An Efficient Formation Flight Simulator with Extensions to Unsteady Maneuvers

A THESIS

SUBMITTED TO THE FACULTY OF THE

UNIVERSITY OF MINNESOTA

BY

Peter John Humbert

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF AEROSPACE ENGINEERING AND MECHANICS

WITH *SUMMA CUM LAUDE* HONORS

Advisor: Dr. Maziar S. Hemati, Assistant Professor

Readers: Dr. Ryan S. Elliott, Professor

Dr. Thomas E. Schwartzentruber, Associate Professor and Director of Undergraduate Studies

May 12, 2018

Abstract

A quasi-steady formation flight simulator is implemented in MATLAB by following a previously published methodology for calculating aerodynamic influence among lifting surfaces. The methodology extends Prandtl's lifting-line theory both to multiple lifting surfaces and to lifting surfaces with complex geometries. The simulator presented herein is capable of calculating the forces acting on multiple lifting surfaces and on those with various geometric properties, including taper, sweep, and dihedral. A description of the simulator's implementation is offered, as are several demonstrations of its correctness. Numerous examples of its practical uses are then provided.

The quasi-steady model's use of the section lift coefficient then allows it to be hybridized with an empirical Theodorsen model for unsteady pitching, which in turn allows us to formulate an expectation for the manner in which the unsteady, sinusoidal pitching of a leading wing affects a trailing wing. The pitching motion is found to reduce the trailing wing's efficiency appreciably, but the reduction behaves asymptotically. Though not implausible, this result would need to be validated by experiment, offering one of many opportunities for further work.

Computational efficiency is central to both the quasi-steady and hybrid methodologies. The former only depends on the geometry of the formation flight scenario, thereby avoiding calculations at points between the wings, and the latter similarly avoids the the usual requirement of calculating vortex panel dynamics.

Contents

List of Figures

1 Introduction

Energy conservation has recently been the focus of significant attention as organizations and industries seek new methods to reduce the quantities of energy used by their processes. These efforts are the result of both fiscal and environmental incentives for improving energy conservation, and the aviation industry is especially rewarded for improvements in efficiency and their resulting decreases in energy use. This is well-illustrated by the significant reduction in fuel consumption resulting from seemingly insignificant changes, such as American Airlines' shift from paper navigation charts to tablet computers leading to an annual fuel savings of 400,000 gallons or $1.2 million (in 2013) [4]. These savings are small in comparison to the airline's 2013 fuel consumption and expenditure – 2.46 billion gallons and $7.4 billion [9] – but nonetheless signal meaningful reductions, especially when considered in terms of the environmental impact over a span of years. Additionally, the design of winglets has garnered attention from aircraft manufacturers like Boeing, which introduced a new winglet design with the 737 MAX. This new design notably maintains laminar flow, thereby reducing drag [1].

It is common knowledge within the aviation community that flying in formation – usually in the context of migrating geese – can save energy. The aviation industry has long considered the applicability of formation flight to fuel savings, and this applicability deserves an additional review in light of the recent surge in efficiency's popularity. This can be accomplished by simulating and optimizing the influence an aircraft has on the aerodynamics of nearby aircraft.

Several approaches exist for creating such a simulation. In this case, a form of Prandtl's lifting-line theory is used despite it not traditionally being used for multiple bodies. In this vein, the purpose of this project is not only to study formation flight but also to expand upon the current, incomplete understanding of lifting-line theory's applicability to formation flight. A previously published, quasi-steady methodology [10] for calculating the aerodynamic influences among lifting

surfaces is implemented; it is then modified to calculate unsteady aerodynamic effects when the leading lifting surface is pitching sinusoidally.

1.1 Background and Prior Work

A mathematical construct known as a horseshoe vortex is often used in modeling finite wings, and each horseshoe vortex is comprised of three three-dimensional vortex filaments. In the simplest case, the wing is represented by a single horseshoe as shown in Figure 1 [7].

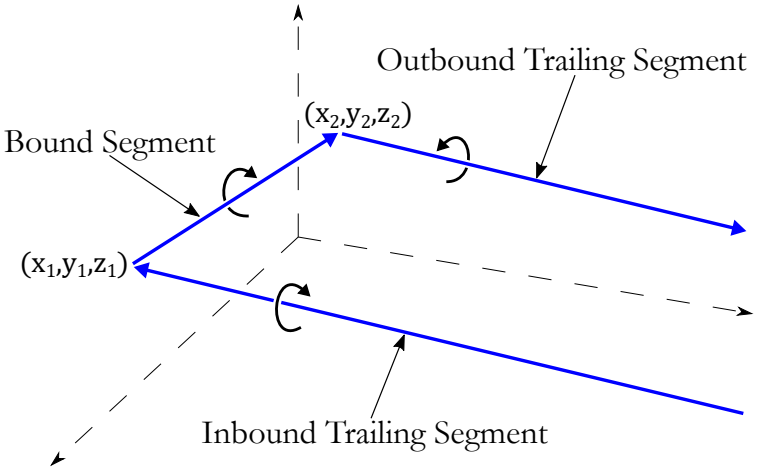
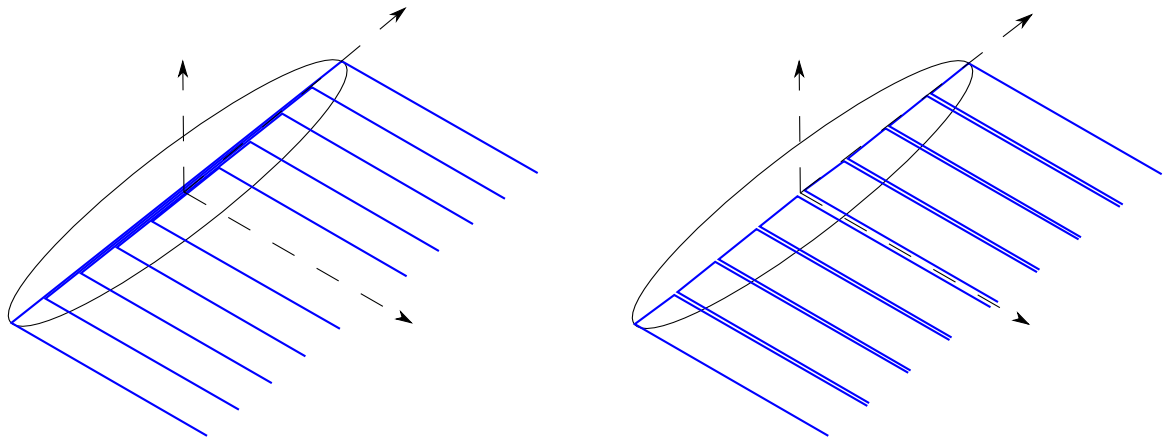


Figure 1: Horseshoe vortex diagram. A horseshoe vortex is comprised of two semi-infinite trailing vortex filaments connected by a bound filament along the lifting surface's quarter-chord line. A fourth vortex segment that connects the two trailing segments, thereby forming a loop, is omitted from the graphic because its infinite distance from the bound segment causes it to have no influence.

Ludwig Prandtl improved upon the one-horseshoe model by using a large number of horseshoes as shown in Figure 2a. This became Prandtl's classic lifting-line theory and is valid for straight wings with aspect ratios $\mathcal{R} > 4$ [11]. Phillips and Snyder [10] further improve upon Prandtl's work by offering a "Modern Adaptation of Prandtl's Classic Lifting-Line Theory" that is far less restricted; it can calculate the forces and moments acting on multiple lifting surfaces with arbitrary camber, sweep, taper, and dihedral. Their model has been validated by both numerical methods and inviscid computational fluid dynamics but avoids the severe computational overhead of each.



(a) Prandtl's original layout of horseshoe vortices. The bound segments would all be collinear along the quarter-chord line, but the illustration shows some streamwise separation to show how the horseshoes are nested.

(b) The layout of horseshoe vortices used in the current work. Interior trailing segments are shown with some spanwise separation but actually overlap, thereby partially canceling each other's vorticity.

Figure 2: Horseshoe vortex layouts.

In numerically solving the strengths of the horseshoe vortices, the local angle of attack, α_i, must also be determined. This value incorporates both the wing's geometric angle of attack and the effect of upwash/downwash at each vortex, and it is used to assign a section lift coefficient, C_{li}, value to each vortex. Brunton and Rowley [3] offer a state-space realization of the Theodorsen pitching model, which allows us to replace $C_{li} = 2\pi\alpha_i$ from thin airfoil theory with a calculation that captures unsteady, two-dimensional effects.

1.2 Contemporary Work

The state-of-the-art work in this field is exemplified by NASA's study of "wake surfing" [2]. At the center of this work is a hardware-in-the-loop, real-time wake simulation that leverages Automatic Dependent Surveillance-Broadcast (ADS-B) to drive an aircraft's autopilot. ADS-B is notably used to gather positional data of the lead aircraft, which is then presented to the trailing aircraft's pilot and will eventually be integrated with the wake estimation. Naturally, this work focuses on the reduction in fuel consumption resulting from formation flight and expands on the prior formation

flight automation in military applications. Both civilian and military studies have resulted in fuel savings of approximately 10%.

Phillips and Snyder's model has recently been applied to the simulation of twisting and flapping wings, like those of birds [5]. Much like the unsteady aspect of the currently presented work, this application seeks to avoid the computational cost associated with unsteady vortex lattice methods.

An emphasis on efficiency has also been seen in the work of Jasa et al. [6], which applies computationally efficient aerodynamic and aerostructural models toward the end of creating software that has instructional value in teaching students about multidisciplinary design optimization. This opportunity is similarly present in the current work, for it could afford students an illustration of the aerodynamics and aerodynamic interactions of finite wings.

Like [5], the flight of insects has also been the object of an application of lifting-line theory. Nabawy and Crowthe [8] assemble a quasi-steady lifting-line model that can tolerate high angles of attack, and the model is then applied to the drag calculation of various insects.

2 The Quasi-Steady Formation Flight Simulator

Although this portion of the simulator simply implements the methodology for calculating lifting surfaces' effects on one another as described by Phillips and Snyder [10], an overview of the simulator's data structures is given in the next section. Further, one notable deviation from their work is described and analyzed. The usage of the quasi-steady simulator is discussed in the next section alongside sample code. The simulator's results are then verified through several methods, ranging from the simplistic interpretation of the the results as being intuitive to the far more rigorous manual calculations. Lastly, practical uses for the simulator are demonstrated.

At an abstract level, the simulator functions with three steps. The user first defines the simulation's geometry. This entails the definition of the lifting surfaces, including position, span, sweep, dihedral, and taper. The induced velocity can be calculated next, followed by the the horseshoe strengths and the local angle of attack at each collocation point. Figure 3 shows how collocation points and the lifting-line are defined. The gray area represents a rectangular wing with non-dimensional chord-length $c = 1$ and aspect ratio $\mathcal{R} = 7$. The lifting-line is placed along the wing's quarter-chord line, and horseshoe vortex vertices are placed along it in accordance with a cosine distribution. The collocation points are then found as the midpoint between each pair of adjacent horseshoe vertices. These points serve as the foundation for the simulation's calculations, for they are the points with which all calculated values are associated.

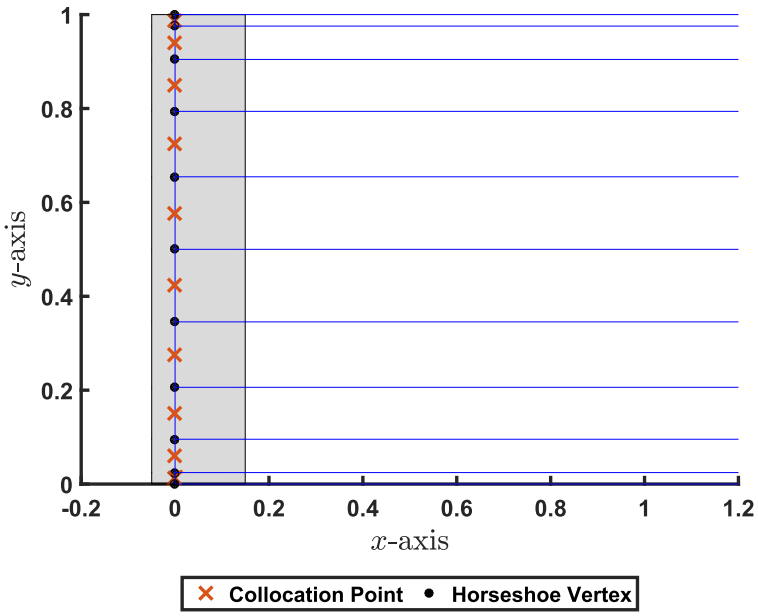


Figure 3: Illustration of a lifting-line as defined in the simulator. The rectangular wing's planform area is shown in grey. The lifting-line is shown in blue, and it is parallel to the y-axis. The semi-infinite trailing vortex filaments are also shown in blue but are aligned with the x-axis. Note that all interior vertices (and their corresponding semi-infinite filaments) are duplicated because adjacent horseshoes overlap each another.

From the vortex strengths, the aerodynamic forces and moments can be calculated, but only the force calculation is implemented in the present work because it provides an ample basis on which to evaluate the efficiency of a given formation flight scenario.

2.1 Notable Implementation Details

A simulation must consist of one or more lifting surfaces, so it follows that one or more `liftingline` objects are required for a simulation. These `liftingline` objects are then encapsulated in a single `simulation` object, which also contains global values, such as the freestream velocity.

Both objects are derived from MATLAB's `handle` class such that their member variables are mutable. This has the distinct benefit of offering better encapsulation. Instead of passing objects to a function and receiving the function's output in the scope in which the object exists, the function's results can simply be added to the objects by the function itself. This eliminates the need for direct

manipulation of the results, which in turn yields more succinct code.

The `simulation` class offers users an additional degree of abstraction from the inner workings of the simulation. For example, calculating the dimensionless vortex strengths, G_i, is accomplished by calling any of the simulation object's functions that rely on G_i (such as `getG` or `getForce`, which retrieve the dimensionless vortex strengths G_i and the force acting on a lifting-line, respectively). The user does not need to interact with any of the underlying `liftingline` objects, leading to another additional benefit: changes made to the mutable `simulation` object can be tracked, so values affected by the changes can be invalidated, leading to their recalculation if the user tries to use them later. For example, calling `getForce` to retrieve force values after modifying the simulation's geometry will automatically result in the recalculation of G_i because the geometry change will invalidate the stored force values. In lieu of the `simulation` object, changes made to the `liftingline` objects would not implicitly invalidate the previously calculated values.

The following functions are required for the simulation's operation, and a brief description of each is provided.

- `findG.m`: for numerically solving the non-dimensional horseshoe vortex strengths, G_i, and the local angles of attack, α_i, which encompass the geometric angle of attack, α_0, and the angle of attack induced by the upwash/downwash from other vortices
- `indvelpll.m`: for assembling the induced velocity matrices
- `RotationMatrix.m`: for the placement of collocation points using Rodrigues' rotation formula, courtesy of Dr. Maziar Hemati
- `C_li.m`: for calculating the section lift coefficient, C_{li}, given α_i
- `getGamma.m`: for converting non-dimensional horseshoe vortex strengths, G_i, to vortex strengths in the direction of the bound segment, Γ

- `findForceNondim.m`: for calculating force vectors from G_i values

A noteworthy deviation from the work of Phillips and Snyder appears in the calculation of the induced velocity. They define the induced velocity matrix $[v_{ij}]$ such that each value is the velocity induced by horseshoe i on collocation point j. The present work dispenses with this definition in favor of the traditional definition of an influence matrix; the value v_{ij} is then the velocity induced by horseshoe j on collocation point i.

The implementation also dispenses with the intuitive structure for storing the induced velocity values: a two-dimensional array for each spatial dimension as in Phillips and Snyder. Instead, a three-dimensional array is used, adding an index k for the lifting line object containing the collocation point with global[1] index i. This has the benefit of simplifying the iteration over the horseshoe vortex vertices, x_1 and x_2. If a 2-dimensional array were used, the algorithm would not simply be able to iterate over a global list of horseshoe vertices pairwise because it needs to identify which horseshoes are the last for their respective `liftingline` objects as well as which `liftingline` object has the next horseshoe. Consider a scenario with 2 lifting-lines, each with 2 horseshoes (as in Section 2.3.4). The first lifting-line will have vortex verticies with indices of 1, 2, and 3; the second will have vertices with indices of 4, 5, and 6. Calculating the induced velocity then requires iteration over the following vertex pairs: $\{1, 2\}$, $\{2, 3\}$, $\{4, 5\}$, and $\{5, 6\}$. Any problem posed by the discontinuity between $\{2, 3\}$ and $\{4, 5\}$ is eliminated with the iteration over the lifting-lines. As such, the inclusion of the lifting-line index can be viewed as a method for enforcing iteration over the lifting-lines.

The notation $[v_{ij}^z]_k$ is adopted for the present work, using the component of induced velocity in the z-direction as an example (hence the superscript z). The subscript k is the index of the lifting-line to which collocation point i belongs, and j is the horseshoe vortex whose influence is being

[1] "Global" refers to the scope of the entire simulation as opposed to the scope of a single lifting-line.

considered. Further, the presence of the subscript k differentiates the current work's induced velocity values from those of Phillips and Snyder. The matrix $[v_{ij}^z]_k$ is `vijz(k,i,j)` in the MATLAB code (`indvelpll.m`), and Algorithm 1 demonstrates how the matrix is assembled.

As shown in Section 2.3.4, the three-dimensional induced velocity matrix implementation can easily be reduced to a two-dimensional matrix. The sum of all n_{ll} induced velocity matrices – one for each lifting line – for a single spatial dimension is equal to the transpose of the matrix $[v_{ij}^z]$ as defined in Phillips and Snyder. Expressing this as an equation yields

$$\sum_{k=1}^{n_{ll}} \left[v_{ij}^z\right]_k = [v_{ij}^z]^T. \tag{1}$$

It should be noted that this is only possible because the calculation of $[v_{ij}^z]_k$ associates the horseshoe indices (in a global context) with the lifting line indices. Recalling the values therefore requires the same association. The induced velocities for the locally i^{th} collocation point of *any but the first* lifting line would be $[v_{(\nu+i)j}^z]_k \mid k > 1$, where ν is the sum of the number of horseshoes comprising the previous lifting lines. Since the matrix already has an index for the lifting line, it would arguably follow that the first collocation point of each lifting line should have index $i = 1$, thereby dispensing with global indices. This, however, would become cumbersome if different lifting lines have different numbers of horseshoes. The result would effectively be a jagged array when assembling the induced velocity matrix for each spatial dimension, not simply a sparse matrix with some fully non-zero rows.

Algorithm 1: The algorithm for assembling the 3 three-dimensional induced velocity matrices. Note that some iterators refer to both mathematical constructs (e.g., horseshoe vortices) and numerical indices; the context should make a given use of an iterator unambiguous.

function indvelpll (P, u)

Input : A 1-dimensional matrix P of lifting line objects and the unit freestream vector u

Output: Three N-by-N matrices ($[v_{ij}^x]_k$, $[v_{ij}^y]_k$, and $[v_{ij}^z]_k$), where N is the aggregate number of horseshoes in the simulation

foreach lifting-line lli in P **do**
  foreach horseshoe ii comprising lli **do**
    $x \leftarrow$ coordinates of the collocation point of ii
    foreach lifting-line llj in P **do**
      $xv \leftarrow$ coordinates of all horseshoe vertices on llj
      $basei \leftarrow$ global horseshoe index offset for lli `// loop omitted`
      foreach horseshoe jj comprising llj **do**
        $basej \leftarrow$ global horseshoe index offset for llj `// loop omitted`

        $x_1 \leftarrow$ coordinates of first vertex of jj `// from` xv
        $x_2 \leftarrow$ coordinates of second vertex of jj `// from` xv
        $r_1 \leftarrow$ vector from x_1 to x
        $r_2 \leftarrow$ vector from x_2 to x
        $mr_1 \leftarrow$ magnitude of r_1
        $mr_2 \leftarrow$ magnitude of r_2

        output $\leftarrow$ Phillips and Snyder Equation 6 for $i = j$, where $i = basei + ii$ is the global index of ii and $j = basej + jj$ is the global index of jj
        if $i \neq j$ **then**
          output $\leftarrow$ sum of *output* and the additional Phillips and Snyder Equation 6 term for the $i \neq j$ case
        end
        output $\leftarrow$ *output* multiplied by the mean aerodynamic chord $\bar{c}$ for ii and divided by 4π

        `/* The following assignments use global indices`
            $i = basei + ii$ `and` $j = basej + jj$ `*/`
        $[v_{ij}^x]_{lli} \leftarrow$ first element of *output*
        $[v_{ij}^y]_{lli} \leftarrow$ second element of *output*
        $[v_{ij}^z]_{lli} \leftarrow$ third element of *output*
      end
    end
  end
end
return $[v_{ij}^x]_k$, $[v_{ij}^y]_k$, $[v_{ij}^z]_k$

The sparsity of $[v_{ij}^z]_k$ warrants a review of its memory complexity. Considering the case of n_{ll} lifting lines each with an average of $\bar{h}$ horseshoes, it is trivial to see that $[v_{ij}^z]$ would have a memory complexity of $\Theta(n_{ll}^2\bar{h}^2)$ and equivalently $O(n_{ll}^2\bar{h}^2)$. This follows from $n_{ll}\bar{h}$ being the total number of horseshoes in the simulation and from the induced velocity matrix being square. For the current implementation, each of n_{ll} lifting lines has one such square matrix. The memory requirement is then $\Theta(n_{ll}^3\bar{h}^2)$, but because of the summative nature of the matrices as given in Equation 1, only $\Theta(n_{ll}^2\bar{h}^2)$ elements are non-zero. The $\Theta(n_{ll}^3\bar{h}^2)$ complexity is both $O(\bar{h}^2)$ and $O(n_{ll}^3)$, the first of which is identical to the complexity of Phillips and Snyder's $[v_{ij}^z]$. However, intuition dictates that the sparsity of $[v_{ij}^z]_k$ results in a markedly greater memory requirement. We therefore conclude that $O(n_{ll}^3)$ is the more descriptive complexity, and the current implementation's memory requirement is a factor of n_{ll} more demanding than the two-dimensional approach.

2.2 Usage

This section offers a step-by-step tutorial for using the simulator. Two identical flat plates with the following attributes are simulated: aspect ratio $\mathcal{R} = 7$, non-dimensional freestream velocity $V_\infty = \langle 1, 0, 0\rangle$, geometric angle of attack $\alpha_0 = 5$ degrees, and number of horseshoes $n = 20$. Corresponding points on the two wings are separated by $\langle 10, 8, 0\rangle$ chord-lengths, thus separating the interior wingtips by one chord-length in the spanwise direction.

An empty simulation object must first be instantiated by specifying the global freestream velocity vector, `Vinf`. Non-dimensionalization by chord-length `c` is advisable to maintain convention. The value of `c` is then unity, and the length scale is universally considered in terms of chord-lengths.

```
c = 1;
Vinf = 1*[1;
          0;
          0];
```

```
sim = simulation(Vinf);
```

The `liftingline` objects need to know the direction of the wake, so the unit vector in the wake direction is calculated from the freestream velocity as `ehat_wake`. The `liftingline` is then given a geometric angle of attack, `alpa0`, and a number of horseshoe vortices, `nshoe`. The endpoints of the leading edge and trailing edge are then defined as `xLE` and `xTE`, respectively, with each point expressed as a column vector. Because these variables define the boundaries of the flat plate, they must take into account the plate's geometric angle with respect to the freestream.

```
ehat_wake = Vinf/norm(Vinf);
alpa0 = 5;
nshoe = 20;

xLE = [0 0;
       0 7;
       0 0];
xTE = [cosd(alpa0) cosd(alpa0);
                 0           7;
      -sind(alpa0) -sind(alpa0)];
```

The first `liftingline` object, `pll_1`, can then be defined. Note that the last argument in the `liftingline` constructor is the variable G_0, a guess of the non-dimensional horseshoe vortex strengths in preparation of the simulator's numerical solving of the actual strengths G_i. The `liftingline` object is then added to the simulation by calling the simulation's `addLL` function.

```
pll_1 = liftingline(xLE,xTE,nshoe,ehat_wake,alpa0,ones(nshoe,1));
sim.addLL(pll_1);
```

The following snippet shows the leading edge and trailing edge variables being overwritten with the values for the second lifting-line, `pll_2`. The lifting-line object is then created and added to

the simulation.

```
xLE = [10 10;
        8 15;
        0  0];
xTE = [10+cosd(alpa0)   10+cosd(alpa0);
                     8               15;
         -sind(alpa0)     -sind(alpa0)];

pll_2 = liftingline(xLE,xTE,nshoe,ehat_wake,alpa0,ones(nshoe,1));
sim.addLL(pll_2);
```

Interestingly, the variable `pll_2` is superfluous; the variable `pll_1` could be overwritten. The simulation must therefore create an internal copy of the `liftingline` objects despite those objects being derived from the `handle` class.

To confirm that the simulation's geometry is correct, the `simulation` object's `plot` function is called as shown in the following snippet and produces the plot shown in Figure 4.

```
sim.plot;
```

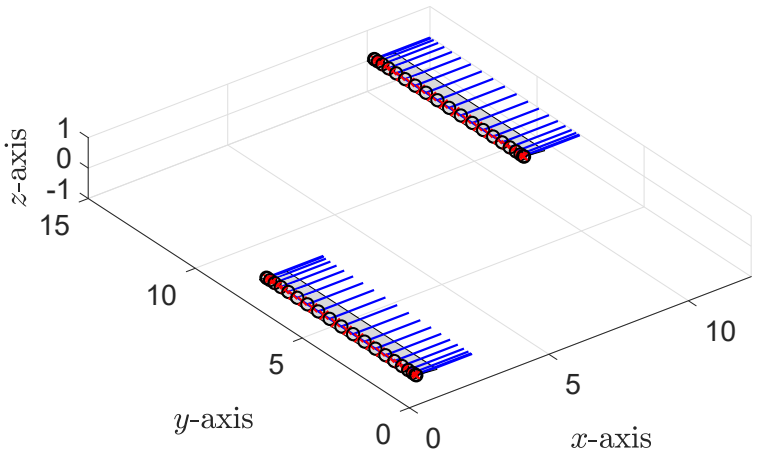


Figure 4: Tutorial scenario geometry. Two identical wings are shown with 10 chord-lengths of streamwise separation and 8 chord-lengths of spanwise separation.

Similarly, the G_i distributions of the two lifting-lines that comprise the simulation can be solved and plotted simply by calling the `simulation` object's `plotAllG` function. The result is Figure 5, which shows both the G_i distributions and the domain covered by the lifting-lines' collocation points (i.e., the blue lines along the spanwise axis).

```
28 sim.plotAllG;
```

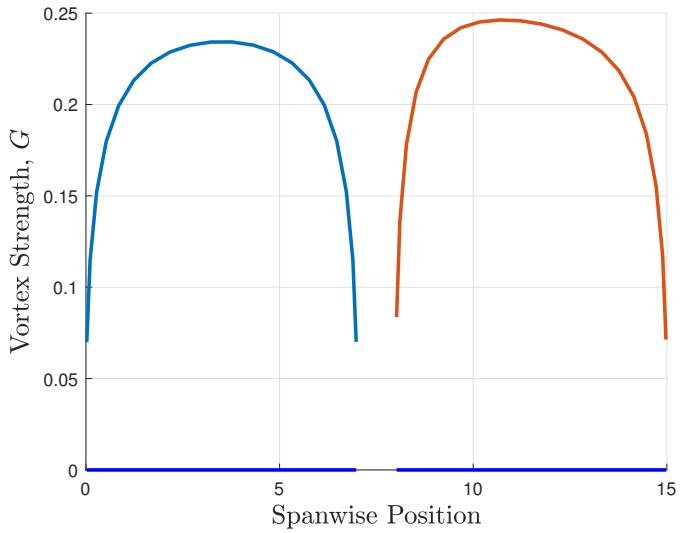


Figure 5: Tutorial G_i distributions. This shows how the non-dimensional vortex strength varies by spanwise position across the two lifting-lines, each of which is represented by a blue line across the horizontal axis.

2.3 Verification

The next two sections show that preliminary verification of the simulation's quasi-steady results is accomplished by simply verifying that the results are physically realistic. The third section takes a more rigorous approach, demonstrating the calculations of a simulation with 2 lifting-lines and 2 horseshoes per lifting-line and showing concordance with the simulation's results.

2.3.1 Infinitesimal Lateral Separation

Two lifting-lines of aspect ratio $\frac{\text{Æ}}{2}$ with infinitesimal spanwise separation and zero streamwise separation should produce a combined G_i distribution that matches that of a solitary lifting-line with aspect ratio Æ.

This is demonstrated in Figure 6. The separation between the two lifting-lines (shown in blue along the spanwise axis) is on the order of 10^{-15} chord-lengths, the magnitude of which nearly matches machine precision. The G_i values associated with a solitary lifting-line are shown as discrete points along the G_i distributions of the two lifting-lines, and the only apparent difference between the two cases occurs at the wingtips, where the solitary lifting-line case benefits from having its collocation points better clustered.

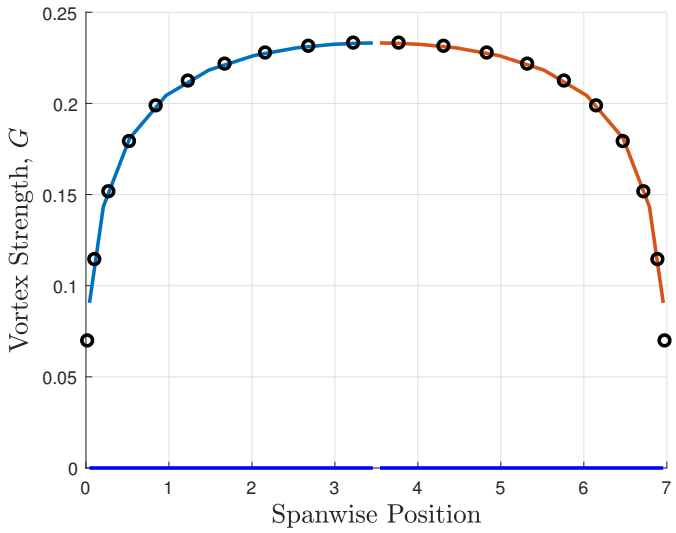
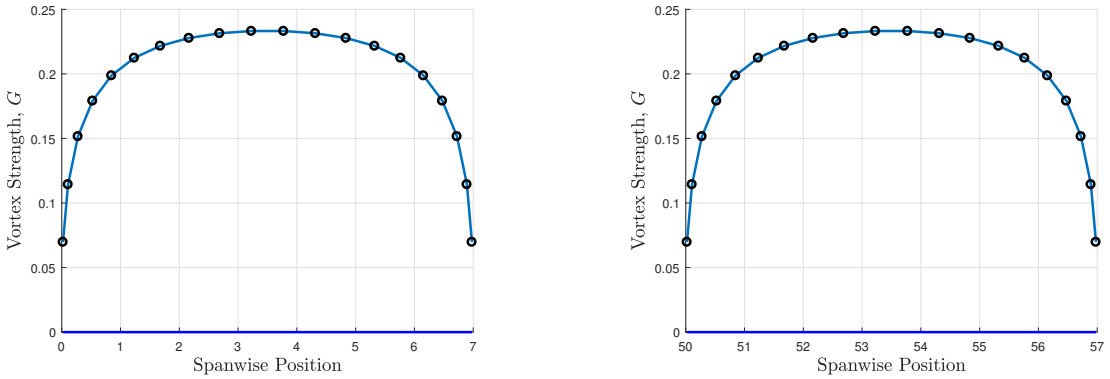


Figure 6: Infinitesimal spanwise separation of two lifting-lines. The G_i values of a solitary lifting-line are plotted as black circles atop the G_i distributions of two lifting-lines separated by an infinitesimal distance. The discrepancy between the two scenarios at the outer ends of the lifting-lines is due to collocation point clustering at spanwise position $y = 3.5$ in the simulation with two lifting-lines, which results in lower resolution near $y = 0$ and $y = 7$.

With the sole discrepancy easily explained, we can conclude that the results match expectations and thereby suggest that the simulator works as intended.

2.3.2 Infinite Lateral Separation

Similar to the case of infinitesimal spanwise separation, infinite spanwise separation should also yield the solitary lifting-line results. Two identical lifting-lines of aspect ratio $\mathcal{AR} = 7$ are simulated with 50 chord-lengths of spanwise separation between corresponding points. Figure 7 shows that this distance is enough to recover the solitary lifting-line results, again allowing us to conclude that the simulator functions properly. As in the previous section, the G_i values associated with a solitary lifting-line are shown as discrete points along the G_i distributions of the two lifting-lines.



(a) (b)

Figure 7: G_i distributions for two lifting-lines with effectively infinite spanwise separation. The two distributions are identical both to each other and to the solitary lifting-line scenario, which is represented by the discrete points.

2.3.3 Convergence of Vortex Strengths

Two lifting-lines are simulated with 10 chord-lengths of streamwise separation and 8 chord-lengths of spanwise separation between corresponding points. With the positions (as shown in Figure 8) of the lifting-lines held constant, the number of horseshoe vorticies comprising each was varied between $n = 5$ and $n = 20$. The objective is to demonstrate that as the number of horseshoes increases, the numerically solved horseshoe strength values converge.

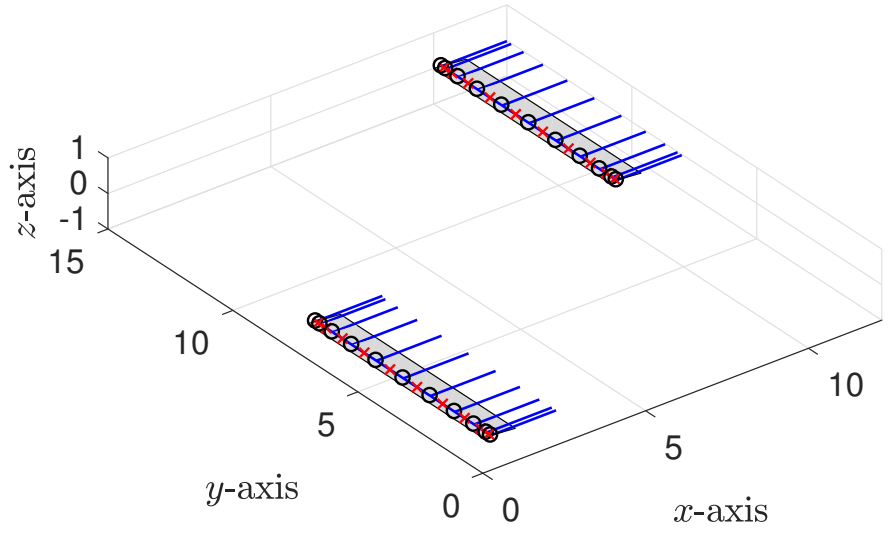


Figure 8: Geometry for the convergence test. This diagram shows the two lifting-lines for the trial of the convergence test that used $n = 10$ horseshoes per lifting-line. Only the number of horseshoes was varied for the test; the positions of the lifting-lines was not changed.

Figure 9 shows that the horseshoe strengths do indeed converge as the number of horseshoes increases. Further, the strengths of the horseshoes near the lifting-lines' midpoints closely agree with their true values (i.e., their values as n increases) even for small n. The significant variation in horseshoe strengths near the ends of the lifting-lines, however, causes poor results for small n and unsurprisingly justifies Phillips and Snyder's recommendation of using a cosine distribution for placing the horseshoes.

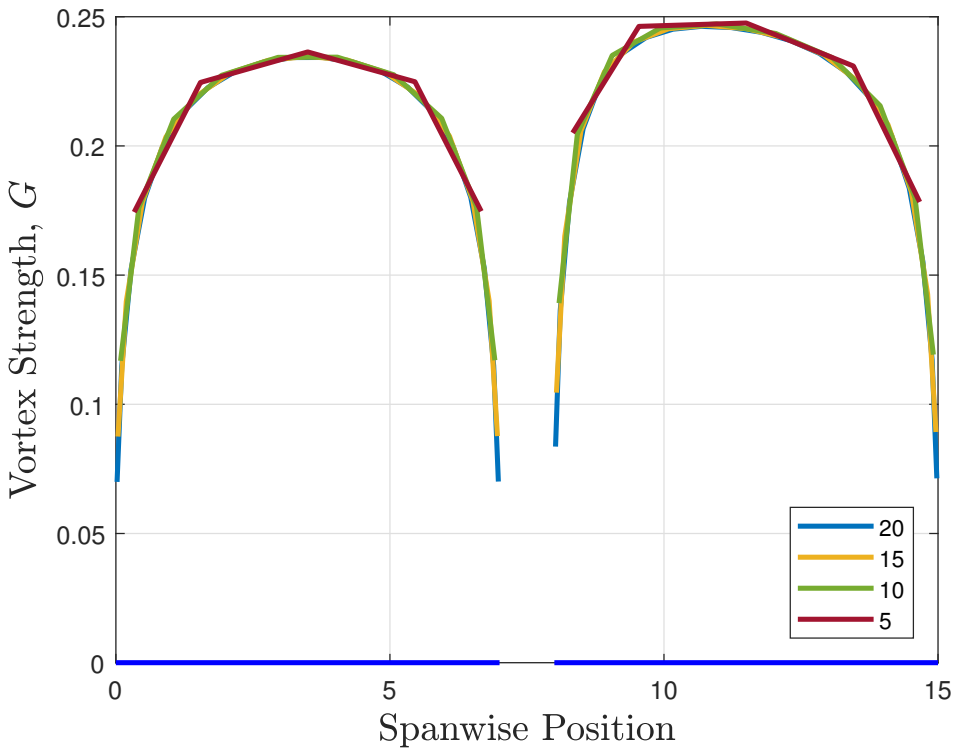


Figure 9: Convergence of vortex strengths. The vortex strength distributions are shown for the two lifting-lines (represented by the blue lines along the spanwise axis) when each lifting-line has $n = 5$, 10, 15, and 20 horseshoe vortices. The distributions rapidly converge, so the accuracy of the simulation strictly increases as the number of horseshoes per lifting-line increases.

2.3.4 Manually Solving the Underlying Equations

To ensure that the manual calculations remain manageable, only two lifting surfaces (with aspect ratio $\mathcal{AR} = 7$) are modeled, and each has only two horseshoe vortices. Corresponding points on the two wings are separated by 10 chord-lengths in the streamwise direction, 8 chord-lengths in the spanwise direction, and 0 chord-lengths in the vertical direction. The geometric angle of attack is $\alpha_0 = 5°$, and the freestream velocity vector is $V_\infty = \langle 1, 0, 0 \rangle$, which is non-dimensionalized by chord $c = 1$. This scenario is illustrated in Figure 10.

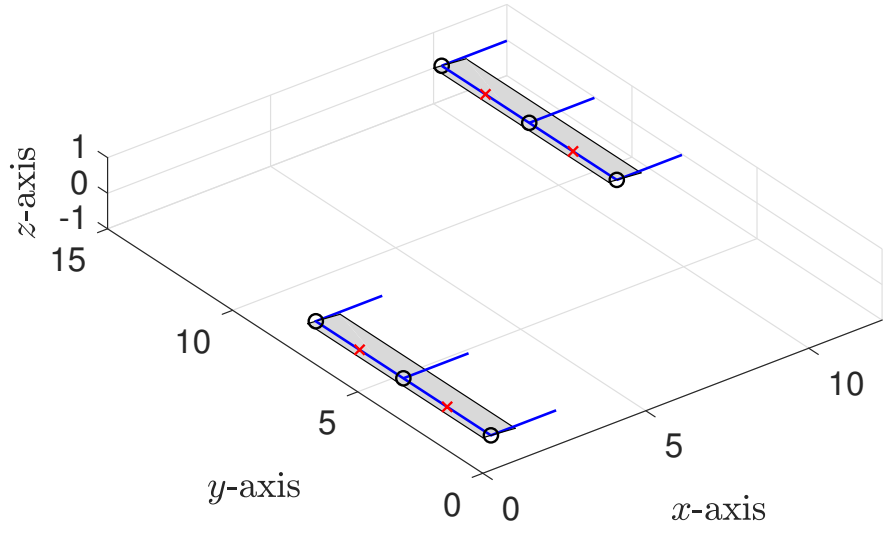


Figure 10: Geometry for manually verifying the quasi-steady implementation. Two identical lifting-lines of aspect ratio $\mathcal{R} = 7$ and geometric angle of attack $\alpha_0 = 5°$ are modeled with two horseshoe vortices each and a separation of $\langle 10, 8, 0 \rangle$ chord-lengths between corresponding points.

Manually solving Equation 6 in Phillips and Snyder results in the following matrix for the z-component of the induced velocity. Note that the matrices for the other spatial dimensions only contain zeros. This is explained by the fact that the two lifting-lines are co-planar, so the rotational motion is strictly vertical at the collocation points. Recall from Section 2.1 that this matrix contains the velocity induced by horseshoe i on collocation point j.

$$\left[v_{ij}^{z}\right] = \begin{bmatrix} -0.09095 & 0.03032 & 0.00819 & 0.00359 \\ 0.03032 & -0.09095 & 0.03120 & 0.00819 \\ 0.00096 & 0.00121 & -0.09095 & 0.03032 \\ 0.00073 & 0.00096 & 0.03032 & -0.09095 \end{bmatrix}$$

Using Algorithm 1, the current implementation produces the following two $\left[v_{ij}^z\right]_k$ matrices:

$$\left[v_{ij}^z\right]_1 = \begin{bmatrix} -0.09095 & 0.03032 & 0.00096 & 0.00073 \\ 0.03032 & -0.09095 & 0.00121 & 0.00096 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\left[v_{ij}^z\right]_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.00819 & 0.03120 & -0.09095 & 0.03032 \\ 0.00359 & 0.00819 & 0.03032 & -0.09095 \end{bmatrix}$$

It is easy to see that these two matrices satisfy Equation 1; that is,

$$\left[v_{ij}^z\right]_1 + \left[v_{ij}^z\right]_2 = [v_{ij}^z]^T.$$

Since the calculations of G_i and α_i require numerical solving, the simulator's output can only be re-inputted into the solved equations to verify the calculations. In particular, Equations 11 and 12 from Phillips and Snyder are combined, and the solved values of G_i are inputted. The resulting values very nearly approach the expected value of zero in being on the order of 10^{-11}. This is also very near the machine precision, which is on the order of 10^{-16}. The global $[G_i]$ vector is found to be

$$[G_i] = \begin{bmatrix} 0.23139 \\ 0.23166 \\ 0.25320 \\ 0.23864 \end{bmatrix}$$

As an example, we consider the expansion of Phillips and Snyder's Equation 11 for the first collocation point ($i = 1$). For the general case of considering horseshoe i, this equation is implemented as

$$2\left|\left(\overrightarrow{v_\infty} + \sum_{j=1}^{N} \overrightarrow{v_{ij}} G_j\right) \times \overrightarrow{\zeta_i}\right| G_i - C_{li}(\alpha_i) = 0. \tag{2}$$

Note that C_{li} can also depend on a control surface deflection, but this relationship was not implemented. Considering that $C_{li} = 2\pi\alpha_i$ for a flat plate and substituting Phillips and Snyder's equation for the local angle of attack, α_i, results in the following:

$$2\left|\left(\overrightarrow{v_\infty} + \sum_{j=1}^{N} \overrightarrow{v_{ij}} G_j\right) \times \overrightarrow{\zeta_i}\right| G_i - 2\pi \tan^{-1}\left[\frac{\left(\overrightarrow{v_\infty} + \sum_{j=1}^{N} \overrightarrow{v_{ij}} G_j\right) \cdot \overrightarrow{u_{ni}}}{\left(\overrightarrow{v_\infty} + \sum_{j=1}^{N} \overrightarrow{v_{ij}} G_j\right) \cdot \overrightarrow{u_{ai}}}\right] = 0, \tag{3}$$

The summation term becomes

$$\begin{aligned}\sum_{j=1}^{N} \overrightarrow{v_{ij}} G_j &= \langle 0, 0, -0.09095\rangle G_1 + \langle 0, 0, 0.03032\rangle G_2 + \langle 0, 0, 0.00096\rangle G_3 + \langle 0, 0, 0.00073\rangle G_4 \\ &= \langle 0, 0, -0.01361\rangle.\end{aligned}$$

The remaining substitutions for Equation 3 are simply dependent on the lifting surface's geometry and the simulation's setup. These substitutions are omitted as it is trivial to demonstrate that the geometry calculations are correct: non-dimensional spanwise length vector $\overrightarrow{\zeta_1} = \langle 0, 1, 0\rangle$, section normal unit vector $\overrightarrow{u_{ni}} = \langle 0.08716, 0, 0.99619\rangle = \langle \sin\alpha_0, 0, \cos\alpha_0\rangle$, and section chordwise unit vector $\overrightarrow{u_{ai}} = \langle 0.99619, 0, -0.08716\rangle = \langle \cos\alpha_0, 0, -\sin\alpha_0\rangle$. The left-hand side of Equation 3 then takes a value of 8.99325×10^{-12}, thereby confirming that the G_i values are correct.

With the G_i values calculated and verified, it is possible to do the same for the aerodynamic forces exerted on the lifting surfaces. Considering the first lifting line ($i \in \{1, 2\}$), the righthand

side of Equation 25 in Phillips and Snyder becomes

$$2\left[(G_1\overrightarrow{v_\infty} + G_1G_1\overrightarrow{v_{11}} + G_1G_2\overrightarrow{v_{12}} + G_1G_3\overrightarrow{v_{13}} + G_1G_4\overrightarrow{v_{14}}) \times \left(\langle 0,1,0\rangle \frac{3.5}{7}\right)\right.$$
$$\left. + (G_2\overrightarrow{v_\infty} + G_2G_1\overrightarrow{v_{21}} + G_2G_2\overrightarrow{v_{22}} + G_2G_3\overrightarrow{v_{23}} + G_2G_4\overrightarrow{v_{24}}) \times \left(\langle 0,1,0\rangle \frac{3.5}{7}\right)\right]$$

This results in the non-dimensional force vector $\langle 0.00628, 0, 0.46305\rangle$, which is identical to the one produced by the simulation.

The accuracy of the implementation of the quasi-steady model set forth by Phillips and Snyder has been comprehensively verified. The induced velocities were first manually calculated, and the resulting values were shown to agree with the simulation's results. The simulation was then used to calculate the G_i values. Equation 11 in Phillips and Snyder was next shown to be satisfied to very high accuracy upon manually inserting the simulation's G_i values into it. Finally, the G_i values were used to calculate the force vector acting on a lifting surface.

2.4 Examples

The next four sections demonstrate the simulator's usefulness in visualizing finite wing aerodynamics and calculating the resulting forces.

2.4.1 Modeling Three Co-Planar Wings in Various Arrangements

Three lifting-lines are simulated in three different scenarios to demonstrate that the results produced by the simulator follow intuition and possess illustrative value.

The first scenario places the three identical lifting-lines in sequence. One lifting-line leads. The second trails the first with a separation of $\langle 10, 8, 0\rangle$ chord-lengths between corresponding points, and the third trails the second with the same separation. This geometry is shown in Figure 11.

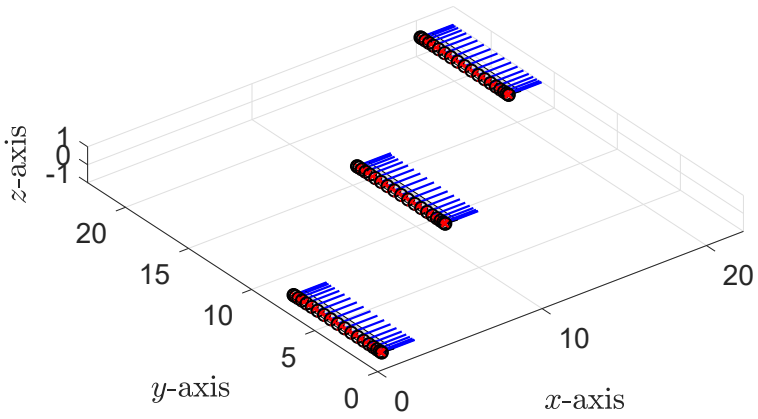


Figure 11: Geometry for modeling three sequenced wings. Each wing is separated by $\langle 10, 8, 0\rangle$ chord-lengths from its neighboring wing(s).

Figure 12 shows the horseshoe strength distribution for the three lifting-lines considered presently and compares the distributions to the distribution for a solitary, identical lifting-line. The second receives a significant contribution from the first. The third receives the same contribution from the second and also benefits slightly from the first, resulting in the third lifting-line having slightly stronger vortices than the second. For both of the trailing lifting-lines, the G_i distribution is skewed toward the leading lifting-line's wake. The influence on upstream lifting-lines by downstream lifting-lines is almost negligible as shown by the very slight increase in strength for the leading lifting-line's vortices.

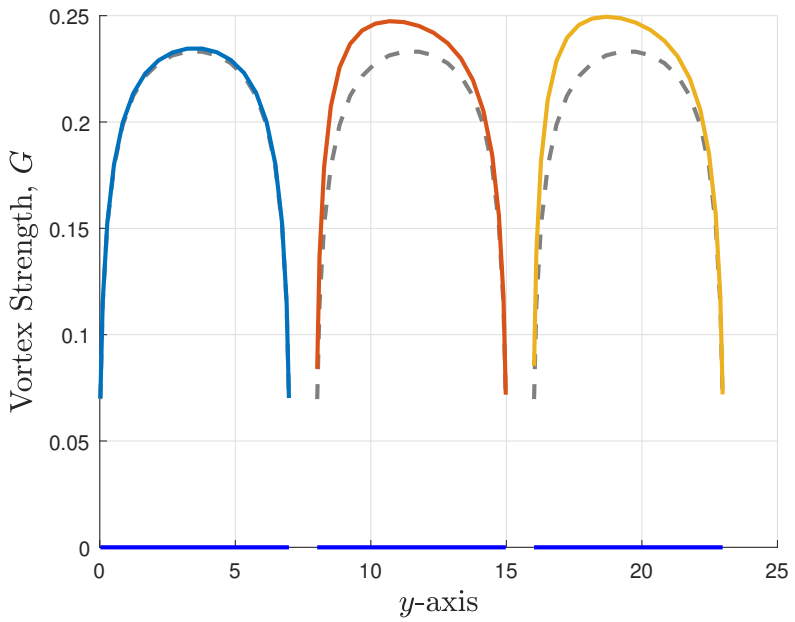


Figure 12: G_i distribution for three sequenced wings. The leading lifting-line receives almost no benefit while the other two each receive a significant benefit from the lifting-line immediately in front of them. The lifting-line that is farthest aft also receives a small, additional benefit from the lifting-line that is farthest forward. The dashed, grey curves show the G_i distribution for a solitary wing.

The next scenario, depicted in Figure 13, involves the typical V-formation with one wing leading and one trailing it on either side.

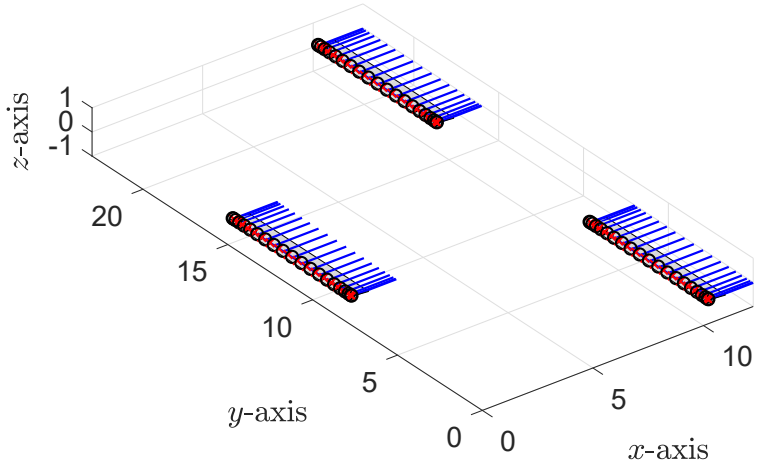


Figure 13: Geometry for modeling a leading wing flanked by two trailing wings. The trailing wings are separated from the leading wing by $\langle 10, \pm 8, 0 \rangle$ chord-lengths.

Figure 14 shows the G_i distributions for the V-formation. The trailing lifting-lines unsurprisingly

receive a significant increase in their vortex strengths from the wake of the leading lifting-line; the G_i distribution for the leading wing is almost unaffected by the trailing wings. Considering the scenario's symmetrical geometry, the results are also unsurprising in that they are symmetrical, and the G_i distributions of the trailing lifting-lines are skewed toward the middle.

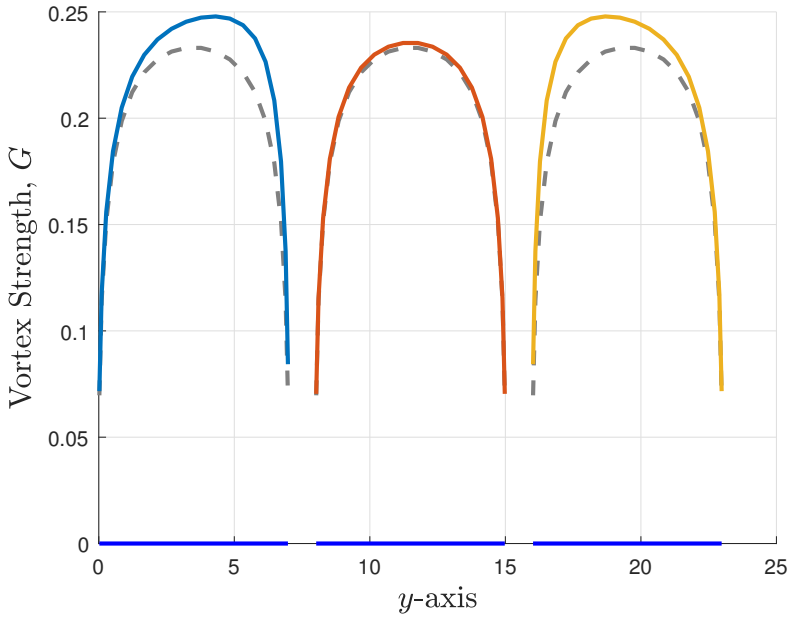


Figure 14: G_i distribution for a leading wing flanked by two trailing wings. The leading lifting-line receives little benefit while the two trailing lifting-lines receive a significant benefit, and the G_i distributions of the two aft lifting-lines are unsurprisingly skewed toward the middle. The dashed, grey curves show the G_i distribution for a solitary wing.

The third and final scenario reverses the previous scenario by placing two wings in the lead with one wing trailing between them. The setup is shown in Figure 15.

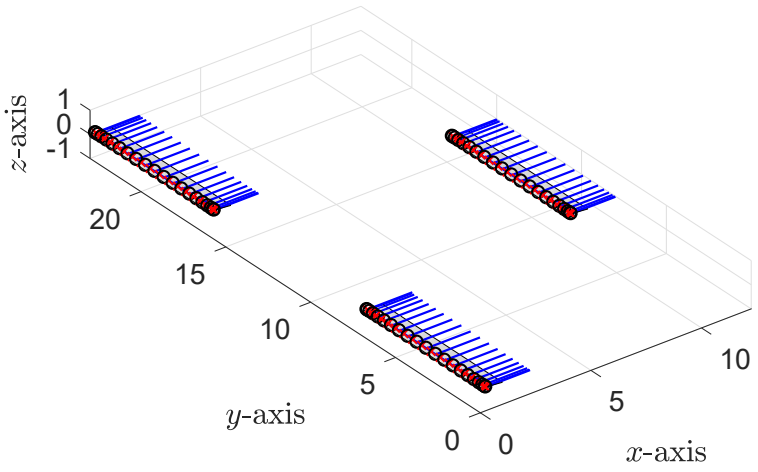


Figure 15: Geometry for modeling a trailing wing flanked by two leading wings. The trailing wing is separated from the two leading wings by $\langle 10, \pm 8, 0 \rangle$ chord-lengths.

Figure 16 shows the G_i distributions for the reversed V-formation. The strengths of the middle lifting-line's vortices are much stronger than those of the flanking lifting-lines, which are largely unaffected. The scenario's symmetrical geometry again yields symmetrical distributions.

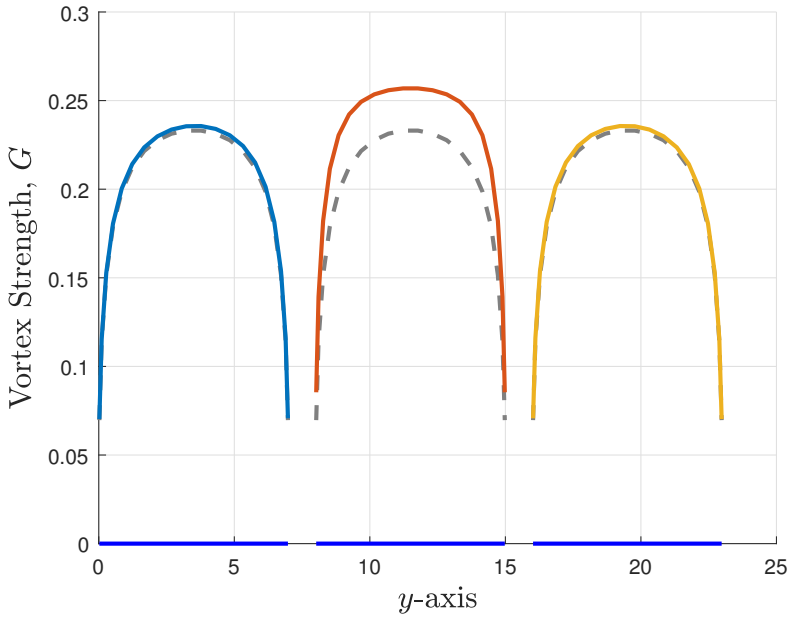


Figure 16: G_i distribution for a trailing wing flanked by two leading wings. The middle lifting-line receives a significant, symmetrical benefit from the wakes of the two leading lifting-lines. The dashed, grey curves show the G_i distribution for a solitary wing.

2.4.2 Modeling a Wing with Sweep and Taper

A trapezoidal wing is modeled with root and tip chord-lengths of 1.5 and 0.5 spatial units, respectively, and the wing spans 7 spatial units. This geometry maintains the usual aspect ratio of $\mathcal{AR} = 7$ such that we can formulate an expectation for the general shape of the G_i distribution. Figure 17 shows the geometry, and Figure 18 shows the G_i distribution, which does not follow the expectation of being a vaguely elliptic curve. Instead, the vortex strengths severely weaken near the root. Phillips and Snyder [10] explain that this is the result of the bound vortex segments producing the most downwash near the root.

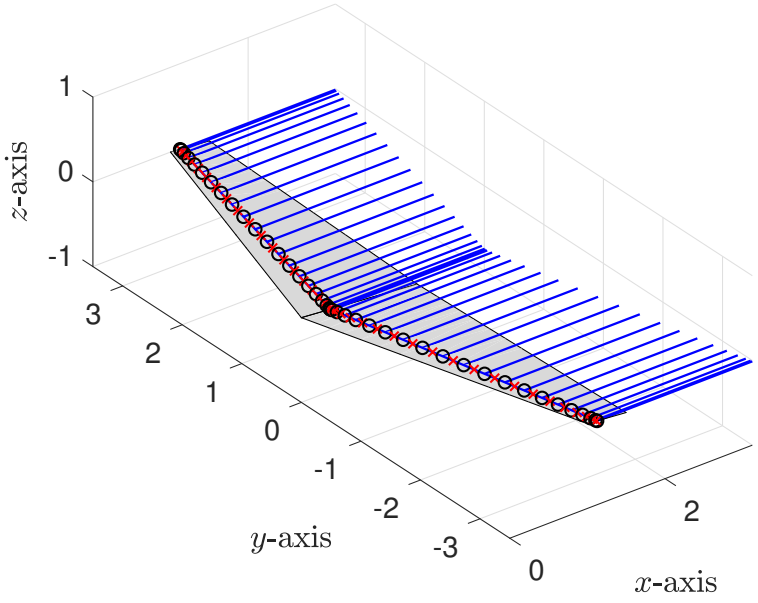


Figure 17: Geometry for modeling a wing with sweep and taper. The wing's root and tip chord-lengths of 1.5 and 0.5 spatial units, together with its span of 7 spatial units, results in an aspect ratio of 7 as has been previously used. The trailing edge is aligned with the y-axis.

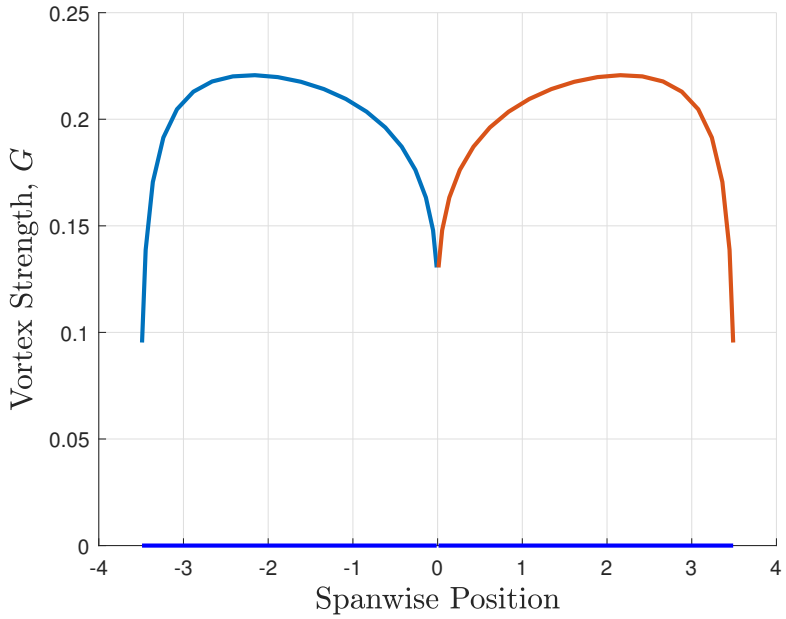


Figure 18: G_i distribution for a wing with sweep and taper. The reduction in vortex strength near the root is explained by the bound vorticity being most influential at this location.

2.4.3 Modeling a Wing with Dihedral and Sideslip

A wing with dihedral can be modeled with two lifting-lines. In this section, a wing with aspect ratio $\mathcal{R} = 7$ and geometric angle of attack $\alpha_0 = 5°$ is considered. The dihedral is included by placing the wingtips one-half chord-length above the root, corresponding to a dihedral of approximately 8.1 degrees, and the sideslip is included by giving the freestream velocity a non-zero y-component. A non-dimensional freestream velocity of $V_\infty = \langle 1.0, 0.1, 0.0 \rangle$ is equivalent to a sideslip angle β of approximately 5.7 degrees. Roll angle is assumed to be negligible. Using 25 horseshoes per lifting-line, this scenario is shown in Figure 19.

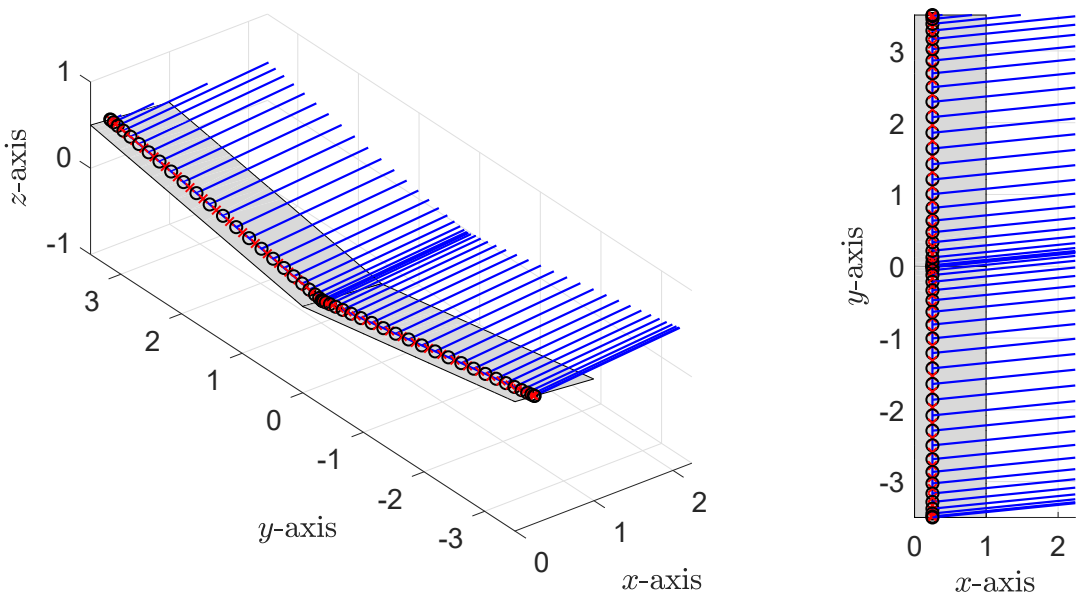


Figure 19: Geometry for modeling a wing with dihedral and sideslip. The graphic on the left shows the dihedral while the graphic on the right shows the angled wake resulting from the sideslip.

We can expect the semi-span that is upwind relative to the y-component of the freestream velocity (i.e., the lifting-line with index 1) to encounter greater lift than it would with $\beta = 0$ degrees, and the opposite is true for the downwind semi-span (with index 2). The distribution of vortex strengths in this case, however, has neither an intuitive nor usefully detailed expectation, but it proves to be reasonable and interesting, as shown in Figure 20. The vortices comprising the upwind semi-span are stronger than those comprising the downwind semi-span, and the distribution is continuous.

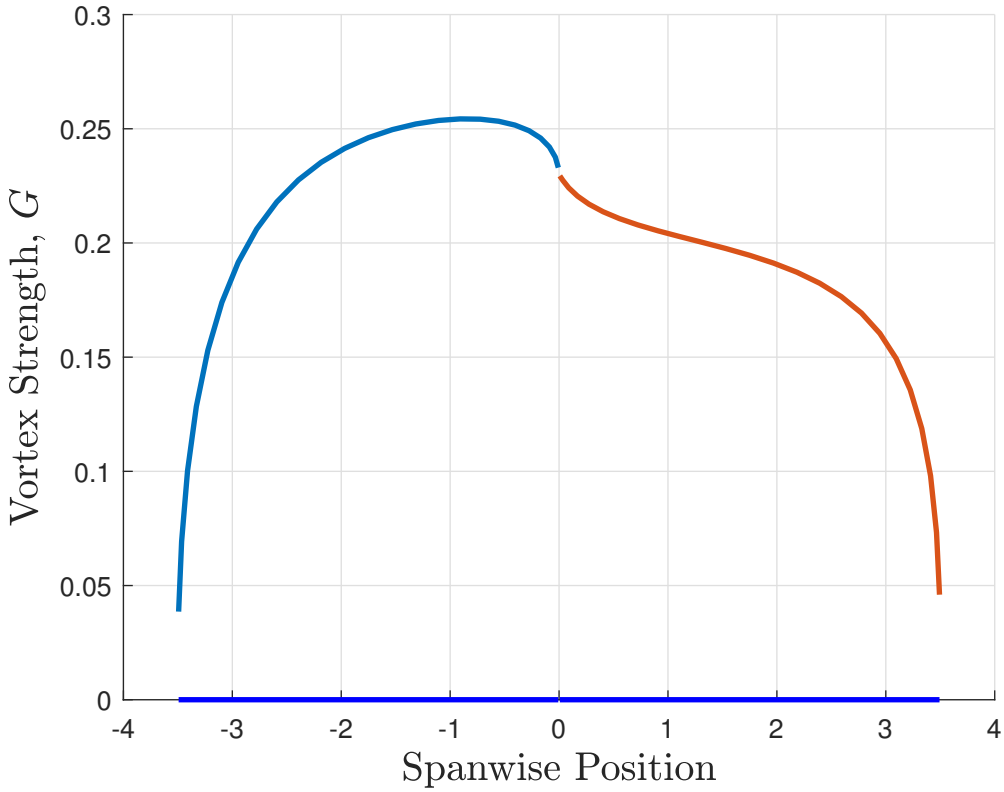


Figure 20: G_i distribution for a wing with dihedral and sideslip. The distribution is notably continuous, and the upwind semi-span generally has stronger vortices than the downwind semi-span. The blue line along the spanwise axis shows the range of spanwise positions occupied by the wing.

From these values of G_i, the forces for the upwind and downwind semi-spans are respectively found to be

$$F_1 = \begin{bmatrix} 0.0054 \\ 0.0638 \\ 0.4466 \end{bmatrix} \quad \text{and} \quad F_2 = \begin{bmatrix} 0.0095 \\ -0.0527 \\ 0.3687 \end{bmatrix}.$$

These results follow the expectation that the upwind semi-span encounters an increase in lift while the downwind semi-span encounters a decrease as indicated by the z-components of 0.4466 and 0.3687 in F_1 and F_2, respectively.

2.4.4 Modeling an Airplane

Using 5 lifting-lines, all three lifting surfaces of a conventional aircraft are simulated simultaneously. The wing has an aspect ratio of $\mathcal{R} = 14$, a geometric angle of attack of $\alpha = 5$ degrees, and non-dimensional chord-length $c = 1$. The horizontal stabilizer has an aspect ratio of $\mathcal{R} = 7$, a geometric angle of attack of $\alpha = 2$ degrees, and non-dimensional chord-length $c = 1$. It is positioned one-half chord-length below the wing to vaguely mimic a Cessna 172. Lastly, the vertical stabilizer extends 1.5 chord-lengths upward from the horizontal stabilizer, and its chord is also $c = 1$. It implicitly meets the freestream at an angle of 0 degrees. Figure 21 shows this scenario.

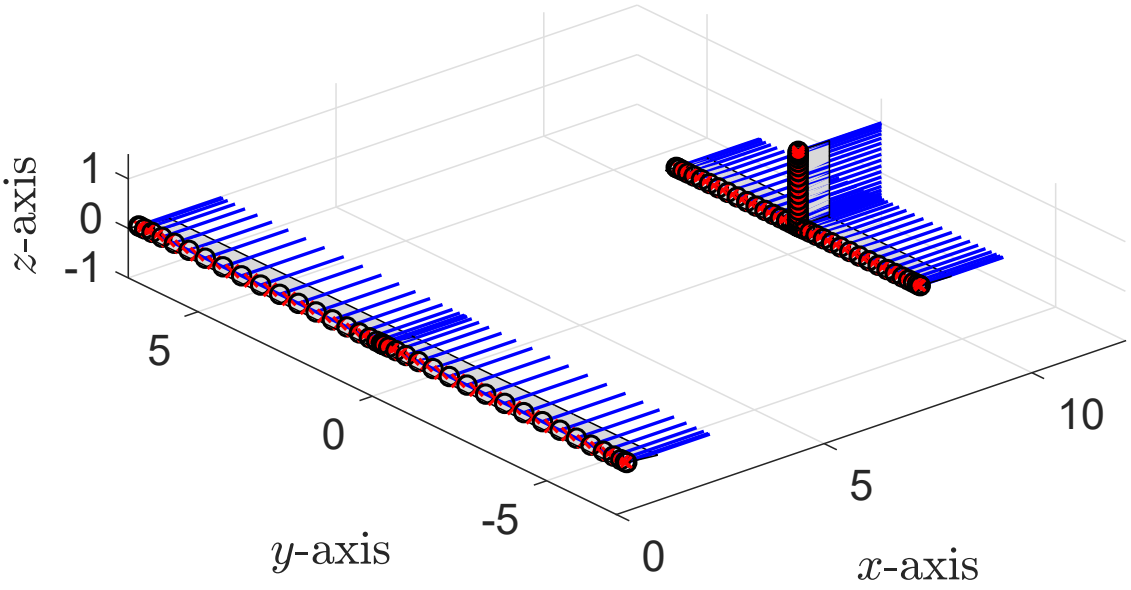


Figure 21: Geometry for modeling an airplane. Using five lifting-lines, we can simultaneously model a wing, horizontal stabilizer, and vertical stabilizer.

The G_i distributions for the horizontal lifting surfaces are unremarkable, but calculating the non-dimensional forces for each lifting surface yields the following reasonable results. The forces acting on the two semi-spans of the wing are F_1 and F_2; the forces acting on the two semi-spans of

the horizontal stabilizer are F_3 and F_4.

$$\overrightarrow{F_1} = \overrightarrow{F_2} = \begin{bmatrix} 0.0054 \\ 0 \\ 0.4684 \end{bmatrix} \qquad \frac{L}{D} = 87.3627$$

$$\overrightarrow{F_3} = \overrightarrow{F_4} = \begin{bmatrix} 0.0018 \\ 0 \\ 0.0942 \end{bmatrix} \qquad \frac{L}{D} = 51.6245$$

There are effectively no forces acting on the vertical stabilizer. This is explained by two simple facts. First, it has no area on which the downwash or upwash from the other surfaces could impinge. Second, it is on the simulation's axis of symmetry, so forces from induced velocity in the x- and y-directions should be balanced.

This scenario also presents an opportunity to demonstrate how the aerodynamic forces are influenced by the horizontal tail's mounting angle. The relationship between the horizontal stabilizer's lifting force and its geometric angle of attack is easily found to be linear. More interestingly, the geometric angle of attack at which the lifting force on the horizontal stabilizer vanishes can be found. For the geometry considered presently, this angle – and equivalently the effective angle of the downwash at the horizontal stabilizer – is 0.864 degrees, which is unsurprisingly greater than zero to counteract the downwash from the wing, and this value can be reached both numerically and by leveraging the linear relationship.

2.4.5 Modeling an Airplane with Dihedral and Sideslip

This section combines the previous two, using a wing composed of two lifting-lines identical to those in Section 2.4.4 but with the wingtips placed one-half chord-length above the root. The horizontal

and vertical stabilizers are unchanged. Sideslip is again added by giving the freestream velocity a non-zero y-component. As in Section 2.4.3, $V_\infty = \langle 1.0, 0.1, 0.0 \rangle$. This geometry, using 80 horseshoes each for the wing and horizontal stabilizer and 20 for the vertical stabilizer, is shown in Figure 22.

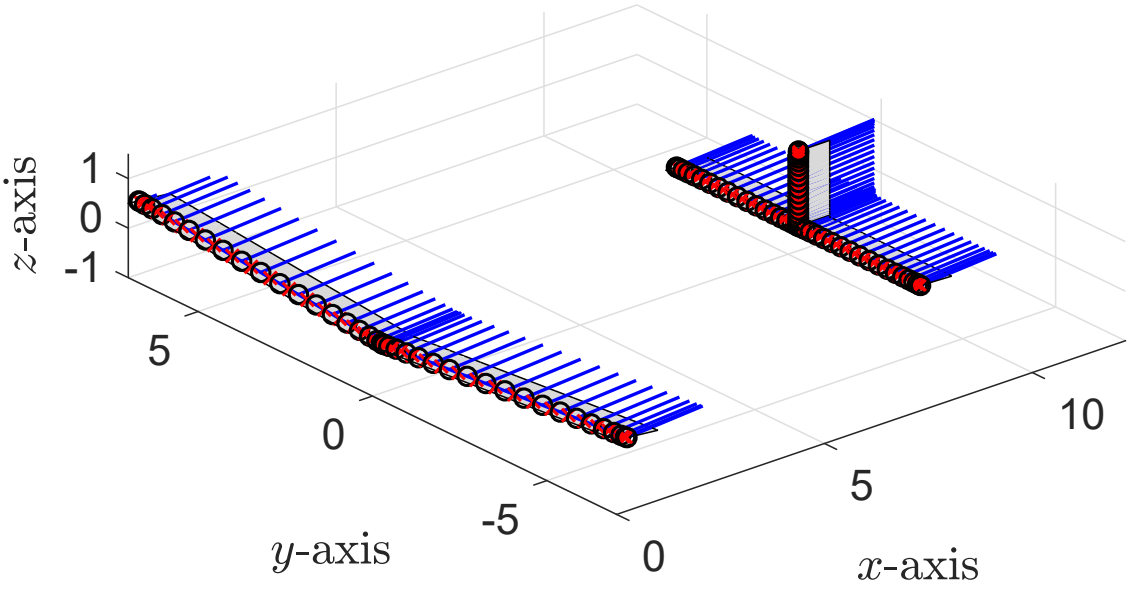


Figure 22: Geometry for modeling an airplane with dihedral and sideslip. This is identical to the geometry for Section 2.4.4 but gives the wing approximately 4.1 degrees of dihedral and a sideslip angle of $\beta \approx 5.7$ degrees.

The G_i distribution for the wing is unremarkable in being qualitatively identical to the one presented in Section 2.4.3, but the distribution for the horizontal stabilizer (shown in Figure 23) deviates from the expectation that it might be similar to the wing. Its G_i distribution also warrants the use for significantly more horseshoes, for there is a sharp inflection near the middle of the span, which might be explained by how the wake of the vertical stabilizer flows over the downwind half of the horizontal stabilizer.

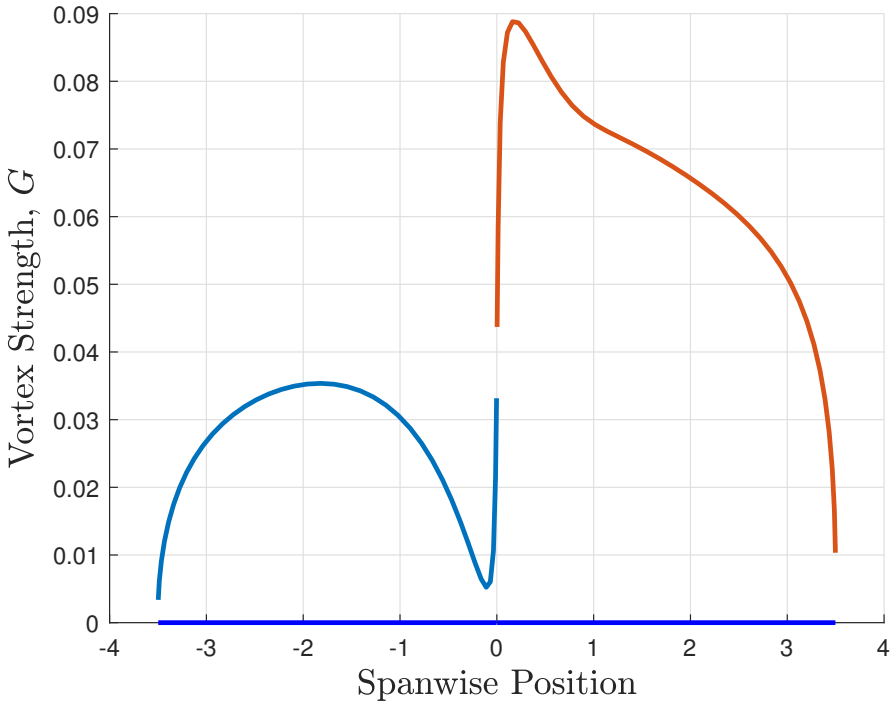


Figure 23: G_i distribution for the horizontal stabilizer of an airplane with dihedral and sideslip. The shape of the distribution is non-obvious but continuous, and the sharp inflection near the midpoint might be due to the vertical stabilizer's non-zero vortex strengths, which in turn result from the sideslip angle.

2.5 Analyses

This section is dedicated to demonstrating the practical analyses that can be conducted with the quasi-steady simulator. The first example considers efficiency as a function of aspect ratio, and the second considers efficiency as a function of wing placement.

2.5.1 Efficiency and Geometric Angle of Attack versus Aspect Ratio

Starting with a rectangular, planar wing with aspect ratio $\text{Æ} = 7$ and non-dimensional chord $c = 1$, the lift force is calculated as 0.4176. This forms the base case. We then vary Æ between 5 and 14 while keeping $\text{Æ}c$ constant at the base case's value of 7 to solve for the geometric angle of attack, α_0, such that the base case's lift force of 0.4176 is maintained. The value then of particular interest

is the efficiency of the wing, expressed as the lift-to-drag ratio, L/D. Figure 24 shows that as $\mathcal{R}$ increases, so does L/D with positive concavity, and α_0 decreases with positive concavity, nearly plateauing at the maximum simulated aspect ratio of 14.

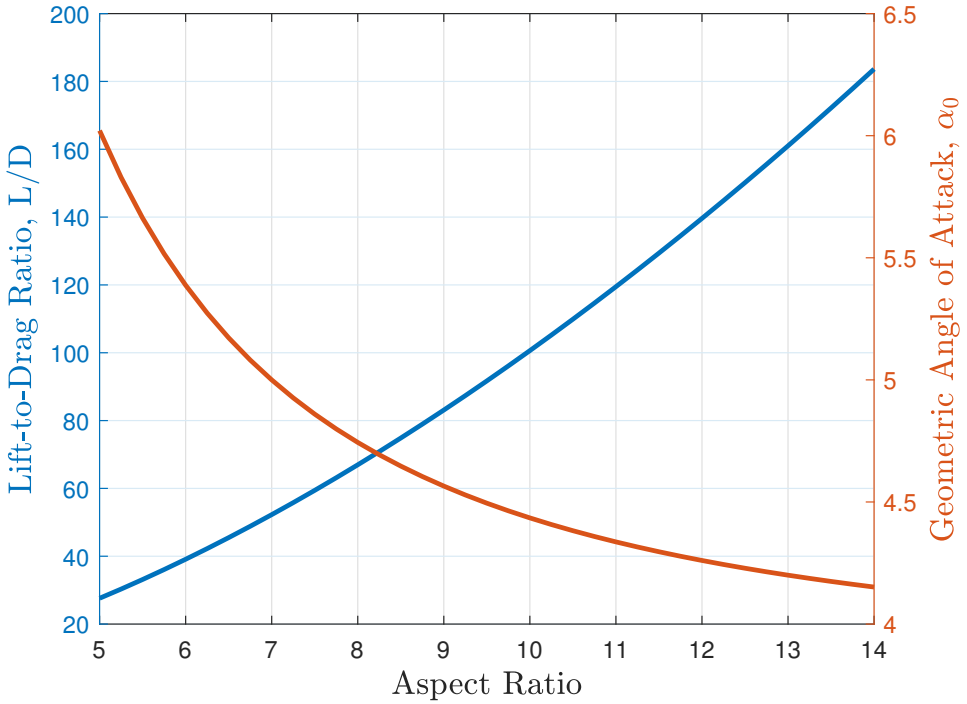


Figure 24: Efficiency and geometric angle of attack versus aspect ratio for constant lift and wing area. As aspect ratio increases, efficiency increases and the geometric angle of attack decreases, and both relationships are nonlinear.

2.5.2 Spatial Lift-to-Drag and Fuel Efficiency Optimization

To see how two wings influence the efficiency of each other, we consider how the lift-to-drag ratios, L/D, of the two wings vary with the position of the trailing wing. Figure 25 shows the setup for this analysis, including the grid of points at which the trailing wing's left-most leading edge point was positioned. The spanwise domain of the grid begins at $y = 10.5$, allowing to wings to always be separated by at least one half-span, and it ends at $y = 15$ such that the lateral separation is not so significant as to completely recover the solitary wing case. The streamwise domain spans $x = 0$ to $x = 25$ in order to show a sizable range of L/D values.

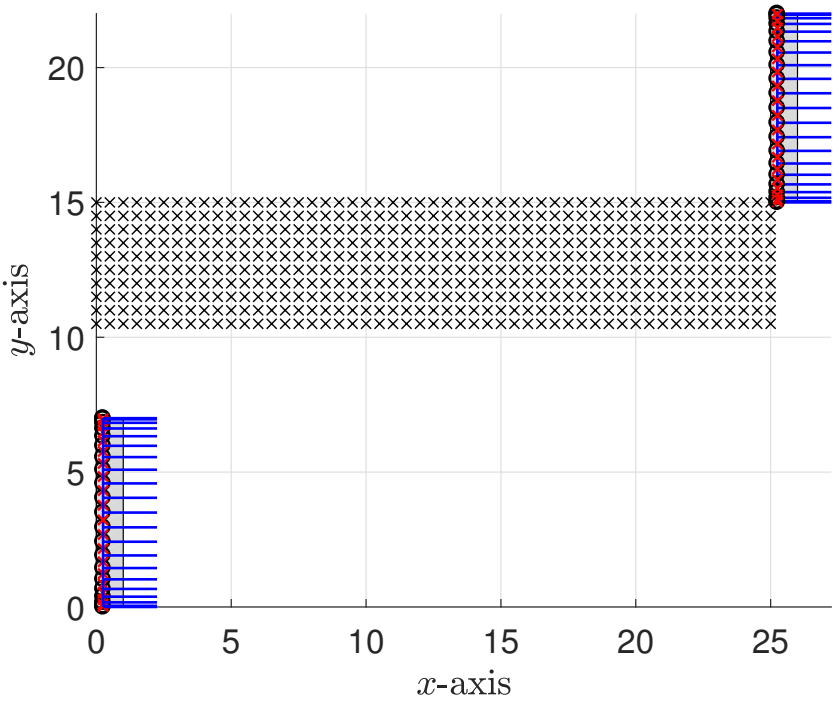


Figure 25: Geometry for spatial lift-to-drag optimization. Each point marked by a black 'X' is the location of the trailing lifting-line's left leading edge point for a single simulation. Together, the points form the grid that is the simulation domain. As an example, the trailing lifting-line is shown with its left leading edge point coincident to the farthest grid point. This was the last trailing lifting-line position simulated.

Using a step size of 0.5 in each of the spatial dimensions resulted in 510 individual simulations being performed. This sizable computational requirement notably prompted the parallelization of the simulations, which requires the data to be manipulated in a particular way. A simplified version of the parallelized code is reproduced below. For brevity, it only aggregates L/D values for the second lifting surface, and these values are stored in `LDresults`. All possible x and y values for the second lifting-line's left-most leading edge point are contained in `possX` and `possY`, respectively. Considering that the `parfor` (parallel `for`) loop iterates over `possX`, it is clear that multiple columns (with constant x) of the simulation domain are solved simultaneously, and each column has its own instance of the variable `tempLDresults` to store its L/D values. The special data manipulation is seen in the use of the iterators. The loops cannot iterate over `possX` and `possY`

directly, instead requiring iteration over dedicated iterators `i` and `j` because direct iteration over the possible coordinate matrices would require incrementing assignments of the dedicated iterators within the loops (e.g., `i = i+1`).

```
step = 0.5;
possX = 0:step:25; ii = 1;
possY = 10.5:step:15; jj = 1;
LDresults = zeros(length(possX),length(possY));

parfor ii = 1:length(possX)
    xbase = possX(ii);
    tempLDresults = zeros(1,length(possY));

    for jj = 1:length(possY)
        ybase = possY(jj);
        % omitted: simulation setup
        tempLDresults(1,jj) = sim.getLD(2);
    end
    LDresults(ii,:) = tempLDresults;
end
```

For the actual study of efficiency's relationship with position, the lift and lift-to-drag values of each lifting-line were calculated and stored. Figures 26 and 27 show the lift-to-drag ratio L/D for the leading and trailing lifting-lines, respectively, as a function of the trailing lifting-line's position. Both of these plots follow the expected trends. For the leading wing, there is little gain in L/D, and the L/D value rapidly nears that of the solitary wing scenario as the trailing lifting-line is moved farther away. The trailing wing receives an appreciable benefit from the leading wing, especially as spanwise separation decreases and streamwise separation increases. The former trend is intuitive. The latter is explained by the fact that the downwash from the leading wing's bound segments

vanishes as streamwise separation increases, but the benefit of the leading wing's trailing segments remains the same.

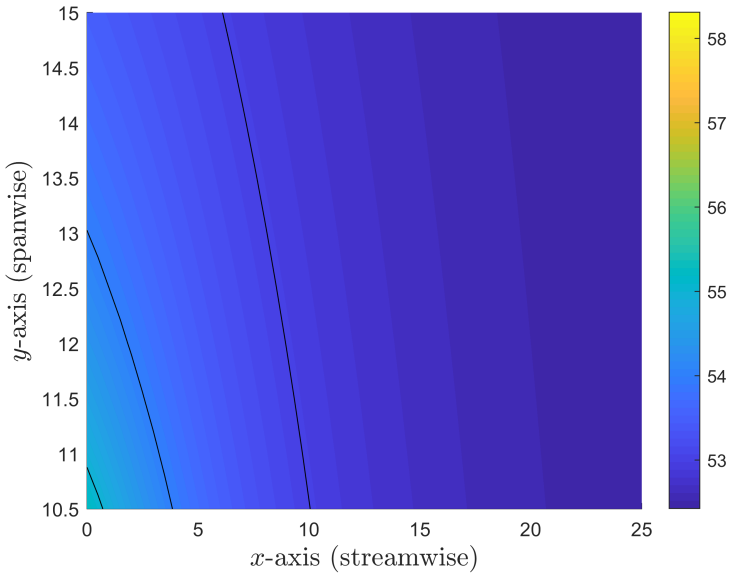


Figure 26: Leading lifting-line lift-to-drag ratio by trailing lifting-line position. As expected, the leading wing receives little benefit from the trailing wing. The L/D values are constant along the black lines, each of which corresponds to one of the color gradient's tick marks.

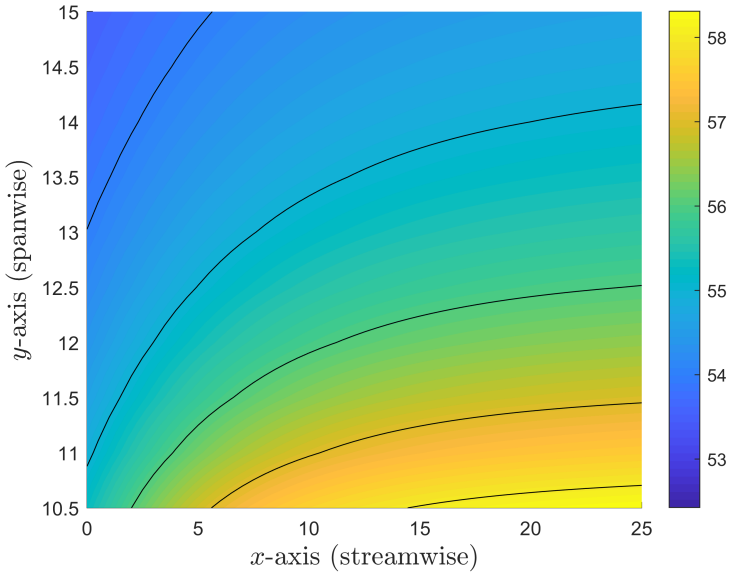


Figure 27: Trailing lifting-line lift-to-drag ratio by trailing lifting-line position. The L/D values are constant along the black lines, each of which corresponds to one of the color gradient's tick marks.

Figure 27 shows one shortcoming of the quasi-steady aerodynamic model. As seaparation becomes infinite, the trailing wing's value of L/D reaches its maximum value for a given amount of spanwise separation. This is illogical because of wake dissipation but provided motivation for the topic of Section 3, which discusses the application of a Theodoresen pitching model to the quasi-steady model.

From the lift force, the lift-to-drag ratio, and the assumptions that the aircraft is flying at a small angle of attack, we can calculate the required thrust of the aircraft as

$$T_{req} = \frac{D}{L}W = \left(\frac{L}{D}\right)^{-1} W, \tag{4}$$

where W is the weight of the aircraft.

If we consider the lifting-lines as representing a pair of jet aircraft, the fuel consumption for each should [11] adhere to the relation

$$\dot{Q} \propto T_{req}. \tag{5}$$

This, however, naively neglects the fact that the force vectors of the two lifting-lines have different z-components. The assumption that the two lifting-lines are associated with aircraft of equal weight would then become invalid, or steady-state flight would not be maintained. In order to maintain the assumption that the aircraft are flying at steady-state, the weight W associated with each lifting-line can be adjusted. It is trivial to see from Equations 4 and 5 that the aircraft's weight simply scales the fuel consumption, and one force's z-component (i.e., the weight associated with one lifting-line) can therefore be used to scale the other.

This is useful in considering that our interest is most focused on the combined fuel consumption

of the two aircraft, i.e.,

$$\left(\dot{Q}\right)_{total} \propto T_{req,1} + T_{req,2} = \left(\frac{C_{L,1}}{C_{D,1}}\right)^{-1} W_1 + \left(\frac{C_{L,2}}{C_{D,2}}\right)^{-1} W_2. \tag{6}$$

By taking $W_1 = C_{L,1}$ and $W_2 = C_{L,2}$ and acknowledging the fact that $C_{L,2}$ will be affected more than $C_{L,1}$ because the second lifting-line is beside the wake of the first, we seek to non-dimensionalize the combined fuel consumption by W_2. The previous expression then becomes

$$\left(\dot{Q}\right)_{total} \propto \left(\frac{C_{L,1}}{C_{D,1}}\right)^{-1} \frac{C_{L,1}}{C_{L,2}} W_2 + \left(\frac{C_{L,2}}{C_{D,2}}\right)^{-1} W_2. \tag{7}$$

The weight W_2 can again be neglected because it simply scales the fuel consumption, effectively resulting in a non-dimensional thrust-specific fuel consumption. To give this value more meaning, we can find the maximum fuel consumption value in the simulation domain and then determine the relative fuel consumption for all other points in the domain. The result is shown in Figure 28. The maximum fuel consumption is associated with the upper left corner, $(x, y) = (0, 15)$, and this is unsurprising because its associated placement of the second lifting-line nearly recovers the solitary wing results. Figure 28 also shows that the domain's minimum fuel consumption is 4.5% less than its maximum.

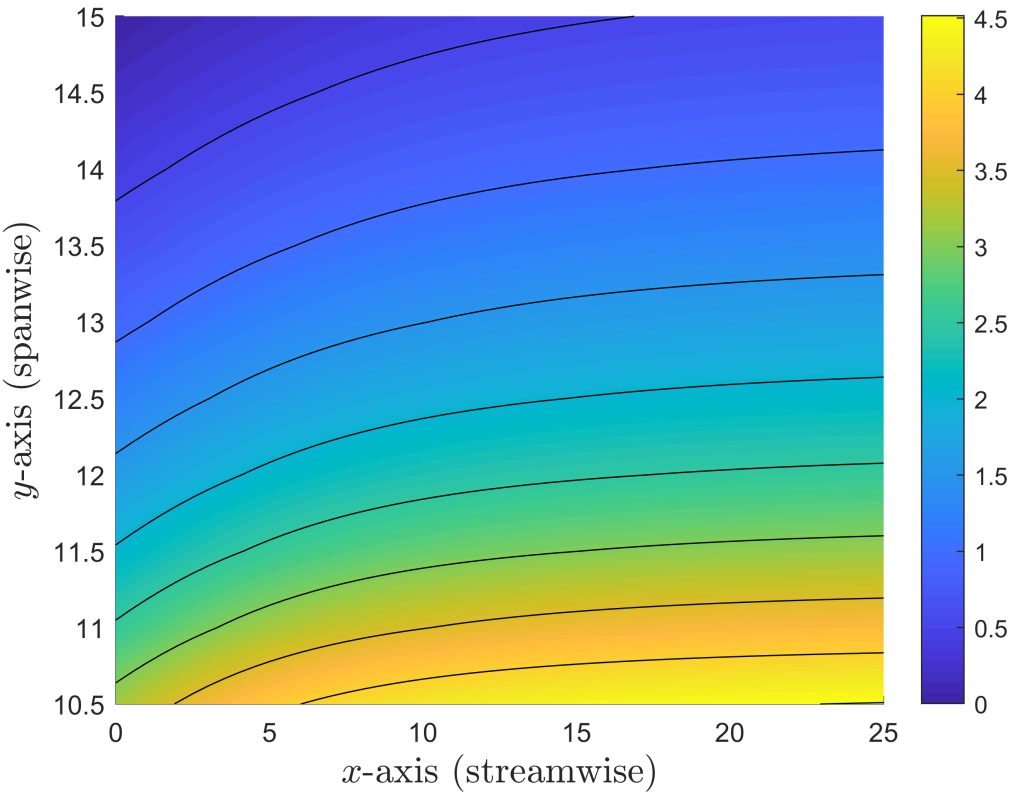


Figure 28: Percent reduction in combined, weight-adjusted fuel consumption for two lifting-lines. This plot shows how the combined fuel consumption varies with the position of the trailing lifting-line. As an example, placing the left point of the trailing lifting-line's leading edge at $(x, y) = (25, 10.5)$ results in a 4.5% decrease in fuel consumption compared to the $(x, y) = (0, 15)$ scenario, which closely approximates a single-wing case. The fuel consumption values are constant along the black lines, each of which corresponds to one of the color gradient's tick marks. The closeness of these lines for low y values indicates that the reduction in fuel consumption quickly decreases as spanwise separation increases.

3 Extending the Quasi-Steady Model for Unsteady Pitching

The quasi-steady model's use of the section lift coefficient allows it to be hybridized with a Theodorsen model. Brunton and Rowley [3] provide an empirical state-space representation for pure pitching motion, pure plunging motion, and combined pitching and plunging motion. This model has been demonstrated to be more accurate than previous work and is implemented in the current project alongside the quasi-steady implementation. Since the Phillips and Snyder model requires local angle of attack to be numerically solved alongside the non-dimensional vortex strengths, we focus our attention on Brunton and Rowley's pure pitching model. The next section describes how this model was combined with the existing model. Section 3.2 provides a tutorial for how to use the unsteady implementation, and Section 3.3 demonstrates the results that it can generate.

3.1 Time-Marching Implementation

Each lifting-line in an unsteady simulation is initialized with the 4-element vector $\widetilde{x}_i = \overrightarrow{0}$ for each horseshoe. A pitching lifting-line is further initialized with the definition of a sinusoid describing how the geometric angle of attack, α_0, varies with time. From the $\alpha_0 = \alpha_0(t)$ expression, $\dot{\alpha}_0(t)$ and $\ddot{\alpha}_0(t)$ must be defined as well. Lastly, the axis about with the pitching occurs is defined in terms of semi-chords from the mid-chord. Setting this value, a, equal to -1 corresponds to pitching about the leading edge.

The first time step can then be simulated. As in the quasi-steady model, the local angles of attack, α_i, and vortex strengths, G_i are numerically solved. The expression for the section lift coefficient in the quasi-steady model, however, is replaced with Brunton and Rowley's balanced truncation model, which is reproduced in expanded form in Equation 8. Note that the time derivatives of the local angles of attack, $\dot{\alpha}_i$ and $\ddot{\alpha}_i$, are assumed to be equal to the time derivatives of the geometric angle of attack, $\dot{\alpha}_0(t)$ and $\ddot{\alpha}_0(t)$, respectively.

$$C_l = \begin{bmatrix} 0.124 & 0.08667 & 0.008805 & 1.156 \times 10^{-4} & \pi & \pi + \pi(1-2a)/2 \end{bmatrix} \begin{bmatrix} \widetilde{x}_i \\ \alpha_i \\ \dot{\alpha}_i \end{bmatrix} - \pi a \ddot{\alpha}_i \qquad (8)$$

With the local angles of attack and horseshoe vortex strengths determined, we can solve for the forces acting on the lifting-lines as in the purely quasi-steady model.

Preparing for the next time step is then accomplished by applying the other half of the balanced truncation model, shown in Equation 9. This requires the same input as the previous part, but α_i is now known.

$$\begin{bmatrix} \dot{\widetilde{x}}_i \\ \dot{\alpha}_i \\ \ddot{\alpha}_i \end{bmatrix} = \begin{bmatrix} -1.158 & -0.3052 & -0.02028 & -2.325 \times 10^{-4} & 2\pi & 2\pi(1-2a)/2 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \widetilde{x}_i \\ \alpha_i \\ \dot{\alpha}_i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \ddot{\alpha}_i \qquad (9)$$

Although this produces the time derivatives of three quantities, we only need to advance $\widetilde{x}_i$ in time. This is accomplished with a first-order forward difference:

$$\widetilde{x}_i(t + \Delta t) = \widetilde{x}_i(t) + \dot{\widetilde{x}}_i(t)\Delta t$$

The next time step can then be simulated. Taken together, Equations 8 and 9 form the state-space realization of Brunton and Rowley's fourth-order balanced truncation model.

3.2 Usage

In addition to the functions and scripts comprising the quasi-steady simulator, two functions are needed for the unsteady extension of the aerodynamic model. These are listed and described below.

- `BruntonCl.m`: for calculating local section lift coefficient C_{li} according to the empirical

Theodorsen model

- `BruntonUpdate.m`: for calculating $\widetilde{x}_i(t + \Delta t)$ from $\widetilde{x}_i(t)$

The setup for an unsteady simulation simply extends the process for creating a quasi-steady simulation. The `simulation` object is defined first, and `liftingline` objects are defined and added to it.

The unsteady state handling is then ascribed to the `simulation` object by passing a timestep to its `setUnsteady` function. Similarly, unsteady pitching behavior is ascribed to a `liftingline` object by calling the `simulation`'s `setPitching` function and passing the following data: the index of the `liftingline` that will undergo pitching; an anonymous function describing its geometric angle of attack as a function of time, $\alpha_0(t)$; the time derivative of $\alpha_0(t)$ as an anonymous function, $\dot{\alpha}_0(t)$; the time derivative of $\dot{\alpha}_0(t)$ as an anonymous function, $\ddot{\alpha}_0(t)$; and the value a. The definition of an unsteady simulation is therefore achieved as follows, with all angular quantities in radians:

```
28 sim.setUnsteady(0.1);
29 sim.setPitching(1, @(t) 0.0175*sin(pi/15*t) + 0.0524, ...
30     @(t) 0.0175*pi/15*cos(pi/15*t), ...
31     @(t) -0.0175*pi^2/15^2*sin(pi/15*t), ...
32     -1);
```

With the requisite geometric information provided to the `simulation` object, the simulation can now be executed. As described in the previous section, this is a two-step task. At each time step, the hybridized model is executed by numerically solving the vortex strengths as in the strictly quasi-steady model. The `getG`, `getForce`, or `getLD` functions of the `simulation` can be used for this purpose. The update portion of the Brunton model is then used to get $\widetilde{x}_i$ for the next time, and the time is incremented. These updates are accomplished by calling the `step` function of the

`simulation`. The following snippet shows an example of the unsteady simulation's execution:

```
N = 500; % number of timesteps to simulate
Forces = zeros(N,3,sim.nlls); % N timesteps by 3 spatial dimensions by
                              % sim.nlls lifting-lines
for ii = 1:N
    F = sim.getForce(1:2);
    Forces(ii,:,:) = F(:,:);
    sim.step;
end
```

3.3 Results

In this section, two identical wings ($\text{Æ} = 7$, $V_\infty = \langle 1, 0, 0 \rangle$, $n = 20$) are modeled. Corresponding leading edge points are separated by $\langle 10, 8, 0 \rangle$ chord-lengths; the trailing edge separation is variable because of the leading wing's variable pitch angle. An approximation of this geometry is shown in Figure 29.

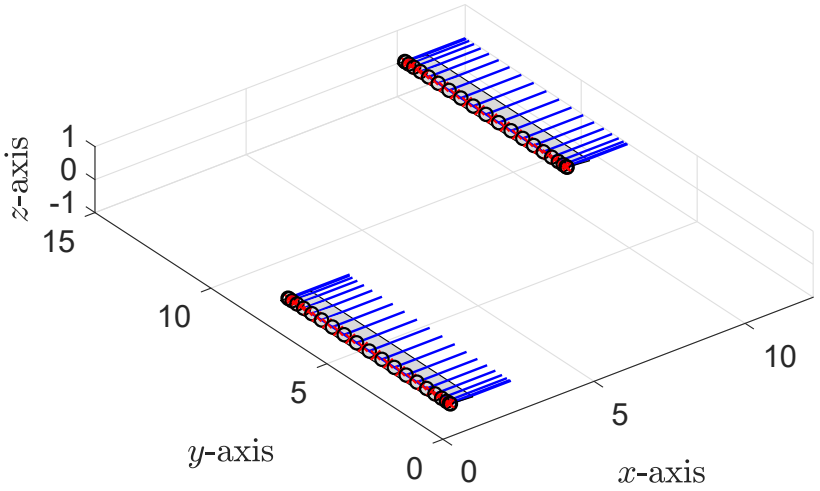


Figure 29: Geometry for unsteady pitching. Two identical wings ($\text{Æ} = 7$, $V_\infty = \langle 1, 0, 0 \rangle$, $n = 20$) are shown with separation of $\langle 10, 8, 0 \rangle$ chord-lengths between corresponding leading edge points.

The leading lifting-line is given the following pitching kinematics:

$$\alpha_0(t) = 0.0175 \sin\left(\frac{\pi}{15}t\right) + 0.0524, \quad \dot{\alpha}_0(t) = 0.0175\frac{\pi}{15}\sin\left(\frac{\pi}{15}t\right), \quad \ddot{\alpha}_0(t) = 0.0175\left(\frac{\pi}{15}\right)^2 \sin\left(\frac{\pi}{15}t\right)$$

These expressions correspond to oscillations between $\alpha_{0,min} = 2^\circ$ and $\alpha_{0,max} = 4^\circ$ with a period of 30 seconds. The objective is to study how the pitching motion of the leading wing affects the efficiency of the trailing wing, and Figure 30 shows that the pitching motion quickly reduces the trailing wing's efficiency. Its efficiency also behaves asymptotically, which suggests that either the effect of the wake's oscillatory buffeting averages over time to a steady state or the hybridized model does not completely capture the unsteady effects for a wing that is not pitching.

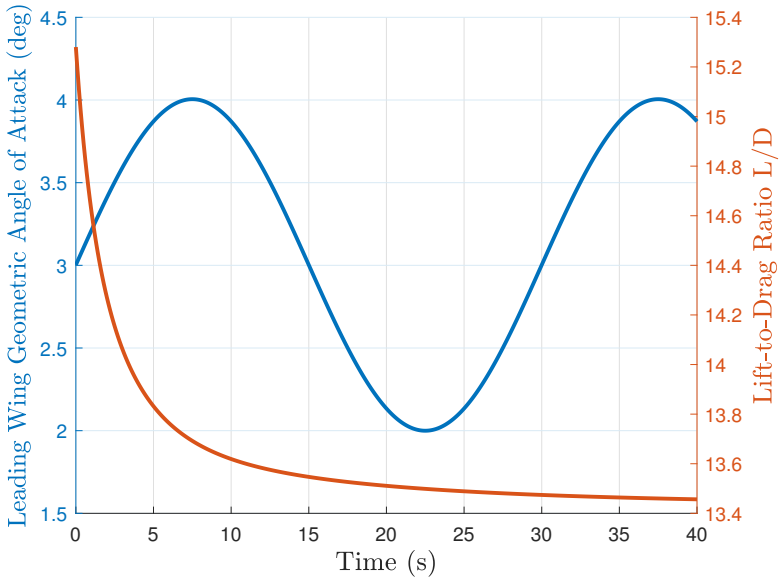


Figure 30: Trailing wing efficiency during leading wing pitching. The efficiency of a trailing wing is significantly decreased by the leading wing's unsteady pitching motion but behaves asymptotically.

4 Conclusions

A quasi-steady formation flight simulator was implemented using a form of Prandtl's lifting-line theory that allows the simulator to calculate the forces acting on multiple wings and wings of various geometries. This versatility is demonstrated by the application of the simulator in studying several formation flight scenarios and geometric properties of wings, including taper, sweep, and dihedral. The simulator is found to be correct and useful for numerous practical studies.

The simulator's aerodynamic model is extended to consider the case of a leading wing undergoing unsteady pitching. Toward this end, an empirical form of Theodorsen's model is applied, and the leading wing's pitching motion is found to decrease the trailing wing's efficiency.

Future work could include the reimplementation of the collocation point placement such that it aligns with the recommendation from Phillips and Snyder that the collocation points be distributed in accordance with the vortices' cosine distribution, not simply placed at the vortices' midpoints. The calculation of moments acting on the lifting-lines could be added, and the quasi-steady sectional lift coefficient $C_{li}(\alpha_i)$ could be modified to depend on flap or control surface deflection. Finally, the results of the unsteady portion of the simulator could be validated by experiment.

References

[1] Boeing. The Most Efficient Winglet on any Airplane. Accessed: 2018-04-28.

[2] Nelson Brown. Wake Surfing: Automated Cooperative Trajectories. Presentation, NASA Armstrong Flight Research Center, 2017.

[3] Steven L. Brunton and Clarence W. Rowley. Empirical State-Space Representations for Theodorsen's Lift Model. *Journal of Fluids and Structures*, 2012.

[4] Susan Carey. IPads Help Airlines Cast Off Costly Load of Paper. *Wall Street Journal*, June 2013.

[5] Jacob S. Izraelevitz et al. State-Space Adaptation of Unsteady Lifting Line Theory: Twisting/Flapping Wings of Finite Span. *AIAA Journal*, April 2017.

[6] John P. Jasa et al. Open-Source Coupled Aerostructural Optimization using Python. *Structural and Multidisciplinary Optimization*, February 2018.

[7] Joseph Katz and Allen Plotkin. *Low-Speed Aerodynamics*. Cambridge aerospace series; 13. Cambridge University Press, Cambridge, UK, 2nd edition, 2001.

[8] Mostafa R. A. Nabawy and William J. Crowthe. A Quasi-Steady Lifting Line Theory for Insect-Like Hovering Flight. *PLoS ONE*, August 2015.

[9] Bureau of Transportation Statistics. Airline Fuel Cost and Consumption (American Airlines - Scheduled) January 1990 - September 2017. Accessed: 2018-04-28.

[10] W. F. Phillips and D. O. Snyder. Modern Adaptation of Prandtl's Classic Lifting-Line Theory. *Journal of Aircraft*, 37(4):662–670, 2000.

[11] Warren F. Phillips. *Mechanics of Flight*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2nd edition, 2010.